ON MATRIX-FREE PSEUDO-ARCLENGTH CONTINUATION METHODS APPLIED TO A NONLOCAL PARTIAL DIFFERENTIAL EQUATION IN 1+1D WITH PSEUDO-SPECTRAL TIME-STEPPING

by

Mitchell Kovacic

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in

The Faculty of Science

Modelling and Computational Science

University of Ontario Institute of Technology

August 2013

Abstract

In this thesis we examine a recent animal aggregation model which describes the evolution of two populations of animals moving on a 1-dimensional spatial domain differing only by the direction they travel. The equations describing the evolution of the populations is a hyperbolic, nonlocal partial differential equation with periodic boundary conditions [5].

We apply pseudo-spectral methods to numerically integrate initial states of the populations given as small perturbations from a homogeneous steady state from which bifurcations and dynamics have been studied from a linear and weakly nonlinear analysis perspective [10, 11]. The existence of transcendental nonlinearities within the equations makes this application of pseudo-spectral methods interestingly nontrivial and simulations do display dynamics similar to those observed in Eftimie *et al.* [5].

Finally we apply matrix-free, pseudo-arclength continuation methods with consideration given to symmetries within the model in an attempt to trace curves from known states to more dynamically exotic regions of parameter space. The flow operator is used to condition the Newton systems arising from the continuation and to allow for a matrix-free continuation algorithm [13]. However, unforeseen degeneracies arise within the Newton system which necessitates further research in order to build a robust continuation software.

Acknowledgements

First and foremost I would like to thank my supervisors, Dr. Pietro-Luciano Buono and Dr. Lennaert van Veen, for their support, guidance, and time. Special thanks goes out for their making of time and promptness as deadlines encroached.

I would also like to thank my parents, Mr. Gary Kovacic and Ms. Theresa Gallant, for financial support, for having a place for me to live without the need to go into debt with rent, and their love and support. I also thank my friends throughout the world whom have given me life experience, comfort, support, and friendships that shall surely extend throughout my life.

Furthermore I would like to thank my colleagues and superiors within the Faculty of Science whom assisted me with friendship, support, and guiding conversations which invariably led to insights I may not have reached without. I also thank the Ontario Ministry of Training, Colleges, and Universities for the Ontario Graduate Scholarship which funded me during the first year of my Masters program.

Lastly I would like to thank University of Ontario Institute of Technology for taking me into their Masters program, for giving me a place to work in comfort, and for offering me opportunities such as teaching assistantships to support me financially.

Author's Declaration

I declare that this work was carried out in accordance with the regulations of the University of Ontario Institute of Technology. The work is original except where indicated by special reference in the text and no part of this document has been submitted for any other degree. Any views expressed in the dissertation are those of the author and in no way represent those of the University of Ontario Institute of Technology. This document has not been presented to any other University for examination either in Canada or overseas.

Mitchell Kovacic

Date: August 21, 2013

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Animal aggregation is the locomotion of animals resulting in pattern formation. Understanding animal aggregation can have serious benefits as the ideas can be applied to pest swarming, human food supply availability, disease transmission, and robotic algorithms [14]. If we better understand the forces and mechanisms that make animals organize themselves then it could help us to notice signs of harmful aggregation and thus be able to take steps to counteract said mechanisms which cause this change. These are the practical reasons for understanding animal aggregation but besides these, Figure 1.1 shows a few examples of different aggregations which are interesting from a mathematical framework. We can identify steep changes in density gradient and what are essentially 1-dimensional curves of animals within 3-dimensional space.

It is clear that understanding how these patterns form is interesting from both a practical and a mathematical framework. In the pursuit of understanding we suggest models that we believe represent approximations to the mechanisms that create these patterns and as with most things we start simple. The prototypical predator-prey models often taught in classes as an introduction to differential equations are examples of such starting places. However these equations only describe the population sizes and does not begin to get into the patterns the animals form.

Discrete, Lagrangian models that simulated each individual within a population under the action of interaction forces were likely the first models introduced that could produce patterns of animal aggregation, analogous to simulations done in physics where particles interacting under electro-magnetic or gravitational forces would be simulated within computers. These Lagrangian models were derived as gradient flows of pair-wise interaction energies describing attraction-repulsion forces. Attraction in

Figure 1.1: A school of fish with a very steep density gradient (left, Taken from http://aquariumprosmn.com/2010/01/460/ July 2nd, 2013 from a posting by Rodney Campbell) and a flock of birds arranged in an essentially 1-dimensional curve (right, Taken from www.seattleaudubon.org/sas/LearnAboutBirds/SeasonalFacts/CanadaGeese.aspx July 2nd, 2013 where it states photo was taken by Russel Link.)

this case is the nature of some living things to congregate with members of its own kind, for protection as an example, while repulsion acts to prevent collisions between members [3]. Morse, an exponential kernel, or Lennard-Jones, a polynomial kernel, are typically used in these attraction-repulsion cases to measure the pair-wise interaction energies between individuals [3, 14]. Note in these cases the interaction kernels have repulsion at close range and attraction at long range.

Observations from data agree qualitatively with these discrete, pair-wise models and these sorts of models became more widely known as swarm dynamics [14, 7]. Eventually continuum models, where the populations are defined as densities through space instead of individuals, were derived from these discrete models in the limit as the number of individuals approached infinity [2]. From here convolutions of these densities of individuals with the interaction kernels became the measurements of attraction-repulsion effects which then affected velocities or turning rates of the populations [14, 7]. Alignment, the coordinated movement of populations which is achieved when individuals react to neighbour movements, became more prevalent in models along with terms to account for restrictive conditions on the way information is received,

such as a limited field of vision [6].

The models introduced by Eftimie *et al.* [5] are the next step in this process of continual refinement. They consider two populations, different only in their direction of travel, living on a 1-dimensional spatial domain. The models include convolution terms with kernels for attraction, repulsion, and alignment forces under five different scenarios for which information can be received and these kernels are different from previous work in that they are Gaussian. Furthermore, the turning rates of these populations is determined by a smooth, monotonic turning function of the interaction forces which fits with observations of turning functions approximated by experiments [2, 7]. This added complexity of the models generates a wide range of interesting dynamics that are not fully understood.

The first reason for this thesis is to attempt to investigate how these dynamics depend on the parameters describing the magnitudes of attraction, repulsion, and alignment forces. The goal is to make qualitative statements like, if you only have large attractive forces then population density distributions tend to be tightly packed, or, if you only have large repulsive forces then population density distributions tend to homogeneous states. These are simple examples that you could perhaps infer from the structure of the equations but we are concerned with more interesting dynamics that are much harder to characterize in such a way; necessitating the use of numerical methods.

The same terms, convolutions and hyperbolic tangent, which seem to generate a wider range of dynamics are also the terms that make these equations harder to deal with numerically. Even worse, the equations do not have a Laplace or similar operator which are known to smooth the solutions and there are symmetries in the model which create degeneracy in the Jacobian of the system of equations. However, what are problems on the surface are actually just chances to learn more and that is the second purpose of this thesis; to apply well known numerical methods, such as pseudo-spectral methods and pseudo-arclength continuation, on atypical problems. We investigate how these difficulties affect these methods and how to fix it.

Chapter 2

The hyperbolic PDE model

2.1 Introduction of model and dynamics exhibited

The models introduced in Eftimie *et al.* [5] describe the evolution of a left-moving population, u^-, and a right-moving population, u^+, with attraction (a), repulsion (r), and alignment (al) interactions. The populations are on a 1-D spatial domain and described as a continuum, thus $u^\pm$ is the density of those individuals at a point and some time. Equation (2.1.1) gives us the evolution, initial conditions, and boundary conditions of these populations,

$$\begin{aligned} \partial_t u^+ + \partial_x(\gamma u^+) &= -\lambda^+(y^+)u^+ + \lambda^-(y^-)u^-, \\ \partial_t u^- - \partial_x(\gamma u^-) &= \lambda^+(y^+)u^+ - \lambda^-(y^-)u^-, \\ u^\pm(x,0) = u_0^\pm(x), \quad u^\pm(0,t) &= u^\pm(L,t). \end{aligned} \tag{2.1.1}$$

Equation (2.1.1) is a hyperbolic, nonlocal partial differential equation in 1+1D. The speed of individuals, γ, may in general depend on space or time but for our purposes it is assumed constant. The length of the spatial domain, L, is taken large in conjunction with periodic boundaries to approximate the positive real line. Note the turning rates, $\lambda^\pm$, are functions of total interaction terms, $y^\pm(u^+, u^-)$, that measure the effects of attraction, repulsion, and alignment which gives us nonlinearity in the model.

Table 2.1 describes how the interaction terms, $y_j^\pm$ for $j \in \{a, r, al\}$, are computed within the five different models. The kernels in these integrals are Gaussian with the form

$$K_j(s) = \frac{1}{\sqrt{2\pi}m_j} \exp\left(-\frac{(s-s_j)^2}{2m_j^2}\right), \quad j \in \{a, r, al\},$$

where $m_j = \frac{1}{8}s_j$; m_j being the width of the interaction kernels and s_j being half the length of the interaction ranges. In general, attraction has the longest range of

Model	Attraction and repulsion
$M1$	$y_{r,a}^{\pm} = q_{r,a}\int_0^\infty K_{r,a}(s)(u(x\pm s) - u(x\mp s))ds$
$M2$	$y_{r,a}^{\pm} = q_{r,a}\int_0^\infty K_{r,a}(s)(u(x\pm s) - u(x\mp s))ds$
$M3$	$y_{r,a}^{\pm} = q_{r,a}\int_0^\infty K_{r,a}(s)u(x\pm s)ds$
$M4$	$y_{r,a}^{\pm} = q_{r,a}\int_0^\infty K_{r,a}(s)(u^{\mp}(x\pm s) - u^{\pm}(x\mp s))ds$
$M5$	$y_{r,a}^{\pm} = q_{r,a}\int_0^\infty K_{r,a}(s)u(x\pm s)ds$
Model	Alignment
$M1$	$y_{al}^{\pm} = q_{al}\int_0^\infty K_{al}(s)(u^{\mp}(x\pm s) - u^{\pm}(x\mp s))ds$
$M2$	$y_{al}^{\pm} = q_{al}\int_0^\infty K_{al}(s)(u^{\mp}(x\pm s) + u^{\mp}(x\mp s) - u^{\pm}(x\pm s) - u^{\pm}(x\mp s))ds$
$M3$	$y_{al}^{\pm} = q_{al}\int_0^\infty K_{al}(s)(u^{\mp}(x\pm s) - u^{\pm}(x\mp s))ds$
$M4$	$y_{al}^{\pm} = q_{al}\int_0^\infty K_{al}(s)(u^{\mp}(x\pm s) - u^{\pm}(x\mp s))ds$
$M5$	$y_{al}^{\pm} = q_{al}\int_0^\infty K_{al}(s)u^{\mp}(x\pm s)ds$

Table 2.1: Form of interaction terms for attraction, repulsion, and alignment forces in the five models. Note that $u(x,t) = u^+(x,t) + u^-(x,t)$

interaction and repulsion has the shortest. Figure 2.1 gives an idea of the location and shape of these kernels.

The total interaction term,

$$y^{\pm} = y_r^{\pm} - y_a^{\pm} + y_{al}^{\pm},$$

is then passed through a turning function,

$$f(y^{\pm}) = \frac{1}{2} + \frac{1}{2}\tanh\left(y^{\pm} - y_0\right),$$

which affects associated turning rates, $\lambda^{\pm}$. Figure 2.2 shows the turning function used as well as the unshifted turning function. If the total interaction term is large and positive then the turning function increases the associated turning rate. Conversely if the total interaction term is large and negative then the turning function decreases the associated turning rate.

With this turning function we form the associated turning rates,

$$\lambda^{\pm} = \lambda_1 + \lambda_2 f\left(y^{\pm}\right),$$

which determine how many individuals change direction. λ^+ (λ^-) is the rate of previously right- (left-) moving individuals turning to the left (right). Figure 2.3 shows the directions in which information can be received to affect λ^+ in the five models. λ^-

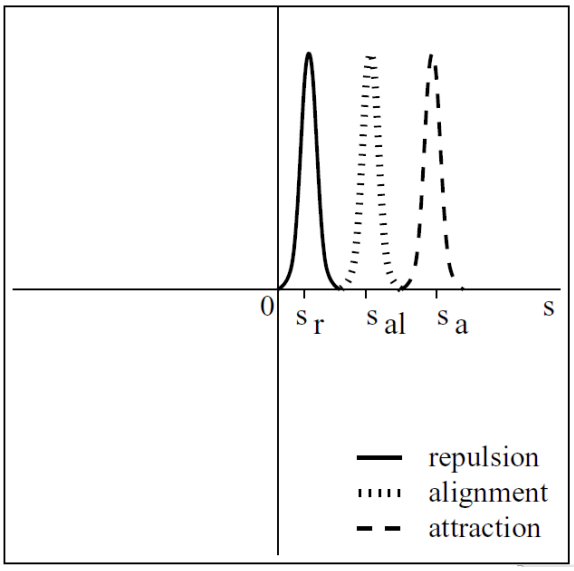

Figure 2.1: Shape and location of Gaussian kernels in interaction integrals. Notice repulsion having its largest influences at very close distances while attraction has its largest influences at longer distances. These distances shown are not properly to scale according to parameter values. Taken from Eftimie *et al.* [11].

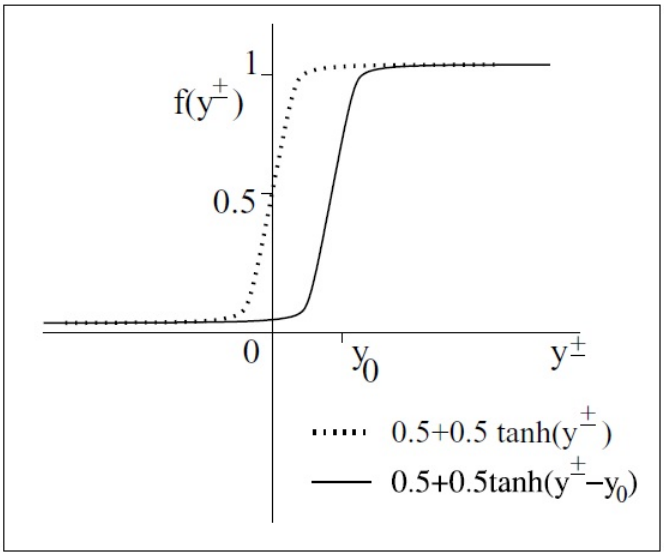

Figure 2.2: Turning function without and with a shift taking the total interaction $y^{\pm}$ as input which is a sum of the attraction, repulsion, and alignment forces. This function determines the turning rates of individuals. Taken from Eftimie *et al.* [11].

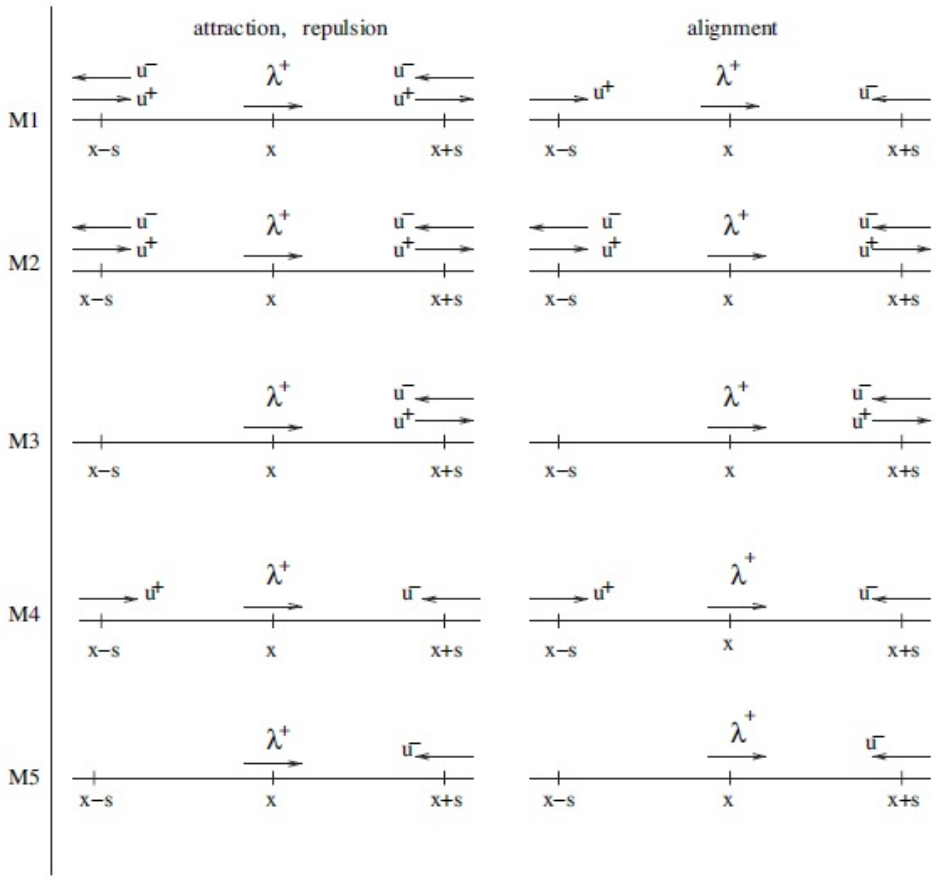

Figure 2.3: A visualization of how λ^+, the rate of originally right-moving individuals turn to the left, receives information from neighbours behind, $x-s$, and in front, $x+s$, of the reference individual at x in the five models. Taken from Eftimie *et al.* [5].

functions in a similarly opposite way.

To better understand the turning rates and their effects we write the turning rates as,

$$\lambda^{\pm} = (\lambda_1 + \lambda_2 f(0)) + \lambda_2(f(y^{\pm}) - f(0)),$$

then the first term represents a random baseline turning rate and the second term represents biasing from the attraction, repulsion, and alignment effects. The shift to the hyperbolic tangent is chosen so the random baseline turning is more accurately approximated by λ_1 and the biasing from interaction influences is more accurately approximated by λ_2.

For our work we focus solely on model M4 and investigate how solutions depend

Parameter	Description	Value
s_a	Half the length of attraction interaction range	1
s_r	Half the length of repulsion interaction range	$\frac{1}{2}$
s_{al}	Half the length of alignment interaction range	$\frac{1}{4}$
m_a	Width of attraction kernel	$\frac{1}{8}$
m_r	Width of repulsion kernel	$\frac{1}{16}$
m_{al}	Width of alignment kernel	$\frac{1}{32}$
λ_1	Approximation to baseline random turning rate	0.2
λ_2	Approximation to bias turning rate	0.9
y_0	Shift of the turning function	2
γ	Speed of individuals	0.1
L	Length of spatial domain	10

Table 2.2: Description and values of fixed parameters of equation (2.1.1).

on the magnitudes of interaction forces, q_j for $j \in \{a, r, al\}$. All other parameters are fixed as shown in Table 2.2.

Even in only one spatial direction Figure 2.4 shows some of the interesting behavior equation (2.1.1) can generate.

2.2 The homogeneous steady state and symmetries of the model

Before we leap into the numerics of evolution and continuation we need to introduce a few key points. Eftimie *et al.* [11, 10] establishes the existence and goes through the linear and weakly nonlinear stability analysis of constant steady states. Because of this, the dynamics about these states are well studied and we will use small perturbations from these steady states as initial conditions. We define these steady states as,

$$(u^+, u^-) = (A^* - c, c), \quad A^* \geq 0, \quad 0 \leq c \leq A^*,$$

where typically we take $A^* = 2$. A^* in this sense is the population at a point so we define the total population on the spatial domain as,

$$A(t) = \int_0^L \left(u^+(x,t) + u^-(x,t)\right) dx.$$

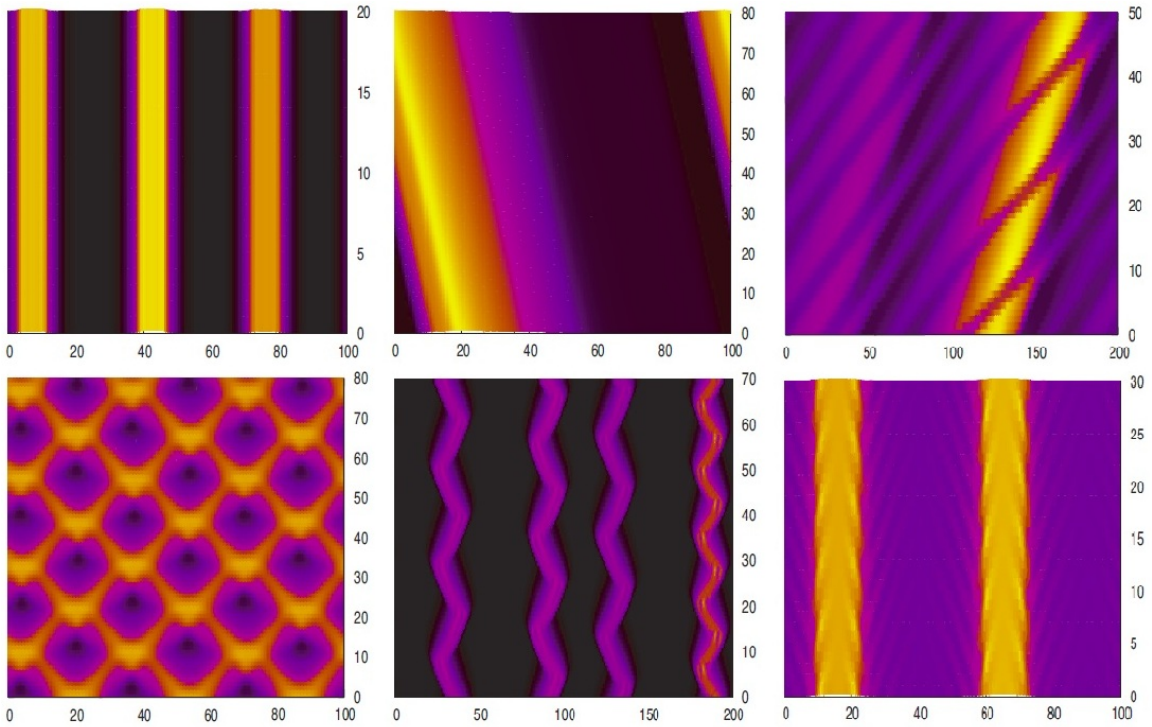

Figure 2.4: Dynamics observed from simulations done by Eftimie *et al.* Plots show the total density of populations at points in space and time. Stationary pulses (top left), traveling pulse (top center), traveling breathers (top right), ripples (bottom left), zigzag pulses (bottom center), feathers (bottom right). Space is along the x-axis with time along the y-axis. Taken from Eftimie *et al.* [5].

Investigating how the total population changes through time we see,

$$\begin{aligned}
\partial_t A &= \int_0^L \left(\partial_t u^+(x,t) + \partial_t u^-(x,t)\right) dx, \\
&= \int_0^L \left(\left(-\partial_x(\gamma u^+) - \lambda^+ u^+ + \lambda^- u^-\right) + \left(\partial_x(\gamma u^-) + \lambda^+ u^+ - \lambda^- u^-\right)\right) dx, \\
&= \gamma \int_0^L \left(\partial_x(-u^+ + u^-)\right) dx, \\
&= \gamma \left(-u^+ + u^-\right)_0^L, \\
&= 0. \qquad \text{From boundary conditions}
\end{aligned}$$

So the total population on the spatial domain does not change with time, thus the initial condition fixes the total population so it is a conserved quantity of equation (2.1.1).

Furthermore it is observed from our simulations that if

$$u_0^+(x) \geq 0, \quad u_0^-(x) \geq 0, \quad x \in [0, L],$$

then

$$u^+(x,t) \geq 0, \quad u^-(x,t) \geq 0, \quad (x,t) \in [0, L] \times [0, \tilde{T}],$$

where $\tilde{T}$ is large. However there is no proof of this at this time.

Returning to the homogeneous steady state we defined earlier, although linear stability analysis was performed for various values of c, we will focus specifically on the homogeneous steady state, $c = \frac{1}{2}A^*$, for our simulations. Figure 2.5 shows curves across which steady state or Hopf bifurcations occur in (q_a, q_r) space with $q_{al} = 0$. These bifurcation curves are points in parameter space where states of the system change stability, therefore these define boundaries between regions in parameter space where different dynamics can be observed from the simulations.

For instance, the region to the lower left corner in Figure 2.5 for which every curve contains the region is the set of parameter values for which the homogeneous steady state is stable. As a parameter crosses a bifurcation curve the homogeneous steady state loses stability and the stability is transferred to another state, possibly with more complex dynamics. An important reason to mention this will be for tests done with the first variational equation since we know parameter values where small perturbations from the homogeneous steady state should decay and the first variational equation

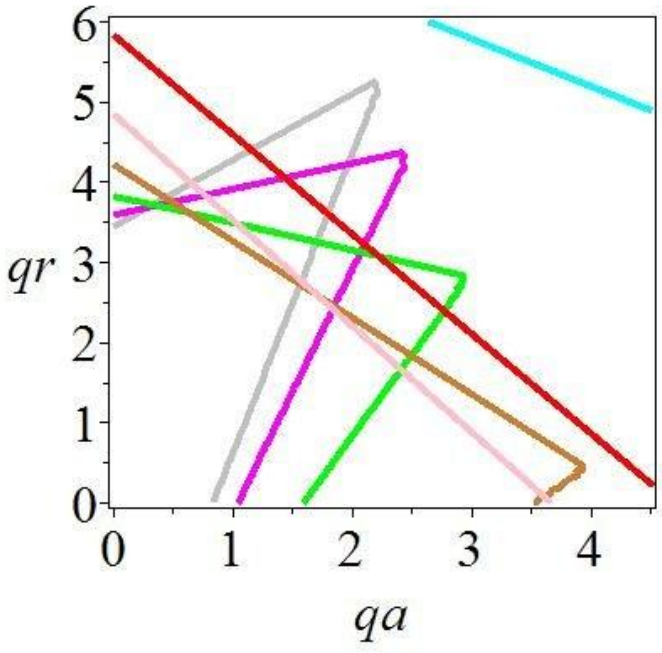

Figure 2.5: Hopf and steady state bifurcation curves in (q_a, q_r) space with $q_{al} = 0$ for M4. The homogeneous steady state is stable for parameter values within the lower left region which is contained by all curves. The different curves represent parameter values over which particular wave numbers of the state bifurcate. Taken from Buono and Eftimie [4].

describes the evolution of perturbations to the solutions. Furthermore, part of our goal will be to determine additional bifurcation points of the system using our continuation software so we may identify regions between potentially more exotic behaviour. For a detailed review of bifurcation analysis and what can be expected from different types of bifurcations, see Kuznetsov [9].

We have one more theoretical property of equation (2.1.1) we need to keep in mind; there is a translation and reflection symmetry under which equation (2.1.1) is invariant. These are a translation symmetry, Θ_y, defined as,

$$\Theta_y u^{\pm}(x,t) = u^{\pm}(x-y,t),$$

and a reflection symmetry, κ, defined as,

$$\kappa\left(u^{+}(x,t), u^{-}(x,t)\right) = \left(u^{-}(L-x,t), u^{+}(L-x,t)\right).$$

See Buono and Eftimie [4] for details of the symmetries.

These symmetries mean if we have a solution, $u(x,t)$, then $\Theta_y u(x,t)$ or $\kappa u(x,t)$ are also solutions. The continuous symmetry, Θ_y, generates a continuous curve of solutions for any given solution called a "group orbit of solutions," except for solutions which are themselves invariant under Θ_y. This symmetry has to be taken into consideration in the continuation of solutions of the system as we will see in Chapter 5, Section 5.2.

2.3 Spatial contraction and problem reformulation

At this point we have the model defined, an understanding of the terms in the model, a homogeneous steady state to use as the initial condition in our evolutions, and definitions of population and symmetry within the model. However, linear terms must be extracted and made obvious from the right-hand side of equation (2.1.1). Additionally we wish to rescale the domain $[0, L]$ to be on $[0, 2\pi]$; this simplifies the use of pseudo-spectral methods later. We also wish to reformulate the problem for simplicity and considerations of storage and numerics later. We restate our full model again before we apply the spatial contraction; our populations evolve according to,

$$\partial_t u^{\pm} \pm \gamma \partial_x (u^{\pm}) = \mp \lambda^{+} u^{+} \pm \lambda^{-} u^{-} \tag{2.3.1}$$

where

$$\begin{aligned}
\lambda^{\pm} &= \lambda_1 + \lambda_2 f(y^{\pm}),\\
y^{\pm} &= y_r^{\pm} - y_a^{\pm} + y_{al}^{\pm},\\
f(y^{\pm}) &= \frac{1}{2} + \frac{1}{2}\tanh(y^{\pm} - y_0),\\
K_i(s) &= \frac{1}{\sqrt{2\pi}m_i}\exp\left(-\frac{(s-s_i)^2}{2m_i^2}\right),\\
y_i^{\pm} &= q_i \int_0^{\infty} K_i(s)(u^{\mp}(x \pm s) - u^{\pm}(x \mp s))ds,
\end{aligned} \tag{2.3.2}$$

with conditions

$$u^{\pm}(x,0) = u_0^{\pm}(x), \quad u^{\pm}(0,t) = u^{\pm}(L,t). \tag{2.3.3}$$

We transform the spatial domain with the following transformations

$$x = \frac{L}{2\pi}y, \qquad w^{\pm}(y,t) = u^{\pm}\left(\frac{L}{2\pi}y, t\right), \qquad m_i = \frac{s_i}{8},$$
$$s = \frac{L}{2\pi}s^*, \qquad ds = \frac{L}{2\pi}ds^*, \qquad s_i = \frac{L}{2\pi}s_i^*.$$

With all this, equation (2.3.1) becomes

$$\partial_t w^{\pm} \pm \gamma^* \partial_y(w^{\pm}) = \mp\lambda^+ w^+ \pm \lambda^- w^-,$$

where $\gamma^* = \frac{2\pi}{L}\gamma$. Equation (2.3.2) becomes

$$\begin{aligned}
\lambda^{\pm} &= \lambda_1 + \lambda_2 f(y^{\pm}),\\
y^{\pm} &= y_r^{\pm} - y_a^{\pm} + y_{al}^{\pm},\\
f(y^{\pm}) &= \frac{1}{2} + \frac{1}{2}\tanh(y^{\pm} - y_0),\\
K_i^*(s^*) &= \frac{4}{s_i^*}\sqrt{\frac{2}{\pi}}\exp\left(-\frac{32(s^* - s_i^*)^2}{s_i^{*2}}\right),\\
y_i^{\pm} &= q_i \int_0^{\infty} K_i^*(s^*)(w^{\mp}(y \pm s^*) - w^{\pm}(y \mp s^*))ds^*.
\end{aligned}$$

Equation (2.3.3) becomes

$$w^{\pm}(y,0) = w_0^{\pm}(y), \quad w^{\pm}(0,t) = w^{\pm}(2\pi,t),$$

where $w_0^{\pm}(y) = u_0^{\pm}\left(\frac{L}{2\pi}y\right)$. With the spatial contraction done we return to the original variable labels to restate our problem in full again as; our populations evolve according

to

$$\partial_t u^{\pm} \pm \gamma \partial_x u^{\pm} = \mp \lambda^{+} u^{+} \pm \lambda^{-} u^{-} \tag{2.3.4}$$

where

$$\begin{aligned}
\lambda^{\pm} &= \lambda_1 + \lambda_2 f(y^{\pm}), \\
y^{\pm} &= y_r^{\pm} - y_a^{\pm} + y_{al}^{\pm}, \\
f(y^{\pm}) &= \frac{1}{2} + \frac{1}{2}\tanh(y^{\pm} - y_0), \\
K_i(s) &= \frac{4}{s_i}\sqrt{\frac{2}{\pi}}\exp\left(-\frac{32(s-s_i)^2}{{s_i}^2}\right), \\
y_i^{\pm} &= q_i \int_0^{\infty} K_i(s)(u^{\mp}(x \pm s) - u^{\pm}(x \mp s))ds,
\end{aligned}$$

with conditions

$$u^{\pm}(x,0) = u_0^{\pm}(x), \quad u^{\pm}(0,t) = u^{\pm}(2\pi,t),$$

and setting

$$\gamma \to \frac{2\pi}{L}\gamma, \qquad\qquad s_i \to \frac{2\pi}{L}s_i, \quad i \in \{a, r, al\},$$

for the proper parameter values on the contracted domain. There are a few things we can simplify from the formulation we have currently. The first is to notice

$$\begin{aligned}
y_i^{\pm} &= q_i \int_0^{\infty} K_i(s)\left(u^{\mp}(x \pm s) - u^{\pm}(x \mp s)\right) ds, \\
&= q_i \int_0^{\infty} K_i(s)\left(\pm u^{-}(x+s) \mp u^{+}(x-s)\right) ds,
\end{aligned}$$

and subsequently notice,

$$y_i^{-} = -y_i^{+},$$

so really we only need to define

$$y_i = q_i \int_0^{\infty} K_i(s)\left(u^{-}(x+s) - u^{+}(x-s)\right) ds$$

so that

$$y_i^{+} = y_i, \quad y_i^{-} = -y_i.$$

Next, it will be easier to compute y_i if the integral is extended over the entire real line so that it is a formal convolution. Eftimie *et al.* [11] explain that the fixed parameters s_i for $i \in \{a, r, al\}$ are chosen in such a way that 98% of the mass of the kernels is within the positive real line so the error introduced by extending the integrals to the whole real line should be insignificant and thus we redefine

$$y_i = q_i \int_{-\infty}^{\infty} K_i(s) \left(u^-(x+s) - u^+(x-s)\right) ds.$$

Next we notice,

$$y^- = y_r^- - y_a^- + y_{al}^- = -\left(y_r^+ - y_a^+ + y_{al}^+\right) = -y^+$$

so again we need only define

$$y = y_r - y_a + y_{al}$$

thus

$$y^+ = y, \quad y^- = -y.$$

We do one final change to the interaction terms; if we redefine

$$q_a \to -q_a$$

then we can form the the total interaction term, y, more generically as

$$y = \sum_{i \in \{a,r,al\}} y_i = \sum_{i \in \{a,r,al\}} q_i (K_i \star u^- - K_i * u^+),$$

where $f \star g$ is the cross-correlation,

$$f \star g = \int_{-\infty}^{\infty} f(s) g(x+s) ds.$$

Next we notice

$$\mp \partial_t u^\pm - \gamma \partial_x u^\pm = \lambda^+ u^+ - \lambda^- u^-$$

is equivalent to equation (2.3.4) and furthermore if we expand the right-hand side we see,

$$\begin{aligned} \lambda^+ u^+ - \lambda^- u^- &= (\lambda_1 + \lambda_2 f(y)) u^+ - (\lambda_1 + \lambda_2 f(-y)) u^-, \\ &= \lambda_1 (u^+ - u^-) + \lambda_2 (f(y) u^+ - f(-y) u^-), \end{aligned}$$

and if we expand again,

$$\begin{aligned} f(y)u^+ - f(-y)u^- &= \left(\frac{1}{2} + \frac{1}{2}\tanh(y-y_0)\right)u^+ - \left(\frac{1}{2} + \frac{1}{2}\tanh(-y-y_0)\right)u^-, \\ &= \frac{1}{2}(u^+ - u^-) + \frac{1}{2}(u^+\tanh(y-y_0) - u^-\tanh(-y-y_0)). \end{aligned}$$

Let us mention this is the last time f will refer to the turning function as we have expanded it out. So our right-hand side becomes

$$\lambda^+u^+ - \lambda^-u^- = (\lambda_1 + \frac{1}{2}\lambda_2)(u^+ - u^-) + \frac{1}{2}\lambda_2(u^+\tanh(y-y_0) - u^-\tanh(-y-y_0))$$

and the linear terms become obvious now. So we restate our full problem one final time for future reference. Our partial differential equation describing the evolution of the populations is given by,

$$\mp\partial_t u^\pm = \gamma\partial_x u^\pm + (\lambda_1 + \frac{1}{2}\lambda_2)(u^+ - u^-) + \frac{1}{2}\lambda_2(u^+\tanh(y-y_0) - u^-\tanh(-y-y_0)) \quad (2.3.5)$$

where

$$y = \sum_{i\in\{a,r,al\}} q_i\left(K_i \star u^- - K_i * u^+\right), \quad K_i(s) = \frac{4}{s_i}\sqrt{\frac{2}{\pi}}\exp\left(-\frac{32(s-s_i)^2}{s_i^2}\right),$$

and

$$u^\pm(x,0) = u_0^\pm(x), \quad u^\pm(0,t) = u^\pm(2\pi,t).$$

To write our differential equation simply as $\partial_t u = f(u)$ for later use we define

$$\begin{aligned} u &= (u^+, u^-)^T, \\ \Lambda &= (\lambda_1 + \frac{1}{2}\lambda_2), \\ f(u) &= \\ &\begin{pmatrix} -\gamma\partial_x u^+ - \Lambda(u^+ - u^-) - \frac{1}{2}\lambda_2(u^+\tanh(y-y_0) - u^-\tanh(-y-y_0)) \\ \gamma\partial_x u^- + \Lambda(u^+ - u^-) + \frac{1}{2}\lambda_2(u^+\tanh(y-y_0) - u^-\tanh(-y-y_0)) \end{pmatrix}. \end{aligned} \quad (2.3.6)$$

Chapter 3
Pseudo-spectral time-stepping

3.1 Motivation

The next step in our work will be to develop an algorithm to time-step initial conditions according to equation (2.3.5) and furthermore to time-step the first variational equation which we will introduce in Chapter 4, Section 4.4. It should be mentioned that the first variational equation will require storage and manipulation of roughly twice the number of variables as equation (2.3.5) so time stepping this will be more costly. This is because the first variational equation follows the state and perturbations from the state, doubling the number of variables when we discretize.

To get a rough idea of the costs of continuation and evolution of the system to a converged state, we show some observed values on our simulations. For each new solution point we wish to compute with our continuation we can expect, from current simulations, around 100 calls to the function which time steps the first variational equation, each with about 7000 time steps. Furthermore, the initial evolution of the perturbed homogeneous steady state to an approximately converged state can take up to 1000000 time steps for some parameter values as there can be slow convergence to states. Figure 3.1 highlights this complicating nature of the slow convergence where solutions that seem to be converged eventually move to another state and require lengthy evolution times to do so.

All the values stated are for our currently implemented pseudo-spectral method, if we were to use finite difference methods in our time stepping we may have needed a smaller time step size and/or finer spatial resolution. This is a reasonable assumption as pseudo-spectral methods are known to achieve good accuracy with a relatively coarse spatial or temporal resolution.

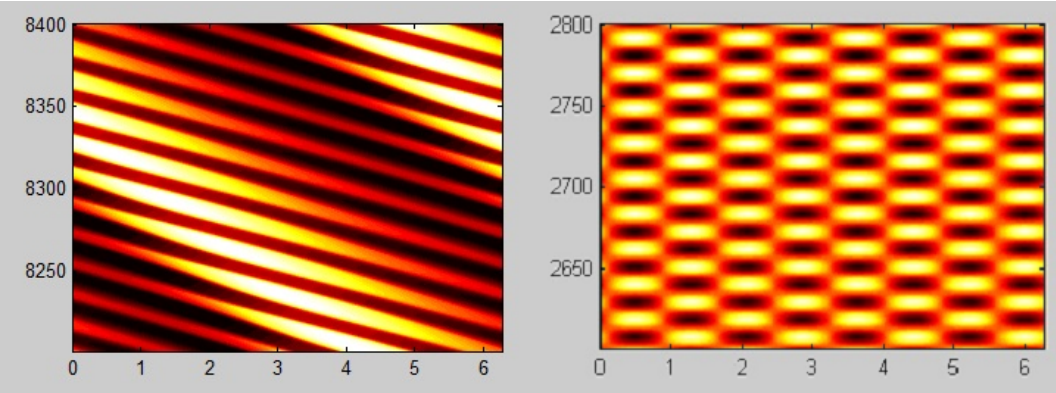

Figure 3.1: Comparison of the same initial condition evolved to two different times. Space is on the x-axis with time on the y-axis. Visualized is the total density of populations at points in space and time.

To get an idea of this difference consider Figure 3.2 which shows a state and three approximations to the derivative. If we use a finite difference approximation of the derivative with $N = 2^{16}$ as a supposed true derivative and compare this to each of the approximations shown we see the pseudo-spectral approximation of the derivative has a global error $\mathbf{O}\left(10^{-4}\right)$ and the finite difference approximation with the same number of grid points has a global error $\mathbf{O}\left(10^{-2}\right)$. To get the same accuracy with finite differencing we need $N = 2^{14}$.

Even if the finite difference methods worked as well with the same time step size and spatial resolution, they would still run slower than our pseudo-spectral methods. With the use of the fast Fourier transform the time taken in solving the discretized system of equation (2.3.5) for a single time-step is reduced to a lower order. The time taken to solve a single time-step of traditional finite differences for this case is $\mathbf{O}(N^2)$ but for pseudo-spectral methods with the use of the fast Fourier transform it is $\mathbf{O}(N \log(N))$. The main point to take away from this is that if the pseudo-spectral methods work they will significantly reduce the time required to evolve an initial condition to a converged state and the time required to do the continuation.

So how well can we expect the methods to work? Based on simulations, for parameter values around the stability region of the homogeneous steady state, we can expect

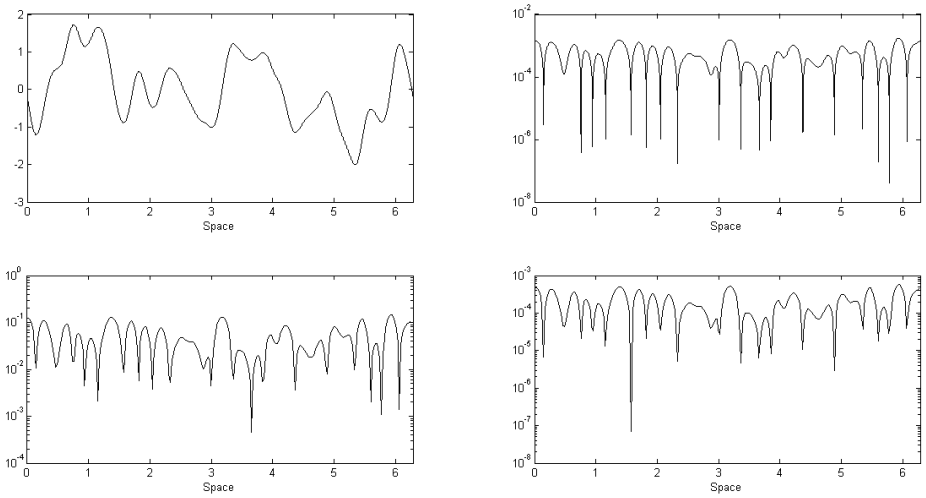

Figure 3.2: A function shown with $N = 2^{16}$ grid points (top left) with the true derivative taken as forward difference approximation. Differences between this benchmark derivative and forward difference approximation with $N = 2^{14}$ (top right), with $N = 2^8$ (bottom left), and with a pseudo-spectral approximation with $N = 2^8$ (bottom right). Global errors of pseudo-spectral approximation with $N = 2^8$ and finite difference approximation with $N = 2^{14}$ are $\mathbf{O}(10^{-4})$ while global error of finite difference approximation with $N = 2^8$ is $\mathbf{O}(10^{-2})$.

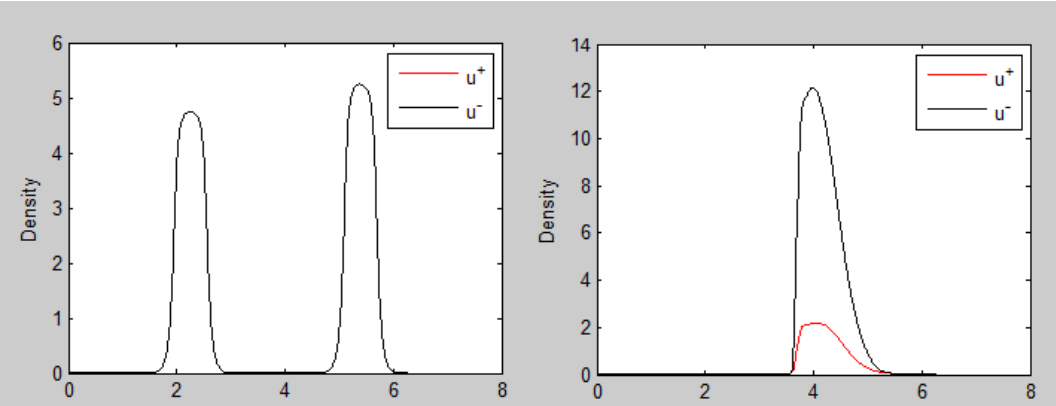

Figure 3.3: Comparison of two solutions evolved with different parameter values, showcasing the possibility of steep gradients for some solutions. Space is on the x-axis and plots show the final time density distribution of populations.

fairly accurate results from pseudo-spectral methods. The basic idea is that if h is your spatial grid size and the solution you are converging to is C^m smooth then you can expect errors between the true solution and your numerical solution to be $\mathbf{O}(h^m)$. Furthermore if the solution you are converging to is C^∞ smooth then one expects super exponential convergence with respect to the grid spacing. So the performance of our pseudo-spectral methods is strongly linked to the expected smoothness of solutions. Figure 3.3 shows us that for some parameter values our solutions seem to be smooth, but for others we get very steep gradients.

Although the solutions are not discontinuous, we unfortunately do not know the extent to how smooth, or nonsmooth, solutions may be. These are issues which will need to be resolved if we wish to continue states beyond the region of stability of the homogeneous steady state as some simulations do show steep gradients and signs of error as parameter values leave the region $(q_a, q_r, q_{al}) \in [0,2] \times [0,2] \times [0,2]$. For further details on pseudo-spectral methods see Trefethen [15].

3.2 The Discrete Fourier Transformation and the Coefficients

Pseudo-spectral methods, in our context, requires us to write out the solution, u, as a sum of space-dependent Fourier basis functions with time-dependent coefficients. We

begin by introducing the discrete Fourier transform (DFT) and the inverse discrete Fourier transform (IDFT) so that we may apply these methods. As stated, we choose a Fourier basis thus let,

$$\phi_k(x) = \exp(ikx).$$

The choice is natural since our problem has periodic boundary conditions. Recall our definition of u from equation (2.3.6). When we use a Fourier basis then the coefficients of the functions will be denoted as $\hat{u}_k$. We choose our spatial discretization as $x_j = \frac{2\pi}{N}j$ with $j = 0, \ldots, N-1$ and therefore we have our function values on the grid as $u_j(t) = u(x_j, t)$. Subsequently we have our DFT,

$$\hat{u}_k(t) = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} u_j(t)\phi_{-k}(x_j), \quad k = -\frac{N}{2} + 1, \ldots, \frac{N}{2},$$

and our IDFT,

$$u_j(t) = \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \hat{u}_k(t)\phi_k(x_j), \quad j = 0, \ldots, N-1,$$

such that they are symmetric since their normalizing constants, $\frac{1}{\sqrt{N}}$ are equal. This is merely preferential but does make some calculations require less book keeping.

When we use the DFT we are transforming from real space to Fourier space. Likewise when we use the IDFT we are transforming from Fourier space to real space. Now what does it mean to be in Fourier space? Fourier space is perhaps more commonly called the space of frequencies and this comes directly from the basis functions. The basis function, ϕ_k, or coefficient, $\hat{u}_k$, is called the basis function, or coefficient, of wave number k. Exactly, the basis function has k crests and troughs. ϕ_0 is the constant function 1 while $\phi_{\frac{N}{2}}$ is a sawtooth function on the grid, bouncing from a crest at one grid point to a trough at the next and continuing one.

An increase in the energy of $\hat{u}_k$, defined as the magnitude of $\hat{u}_k$, causes the amplitude of ϕ_k to increase and thus become more dominant. A plot of the energy in every wave number is then called the power spectrum. Specifically, the power spectrum is a loglog plot with the wave number k on the x-axis and the energy of its associated coefficient $\hat{u}_k$ on the y-axis. Figure 3.4 shows three examples of low, medium, and high

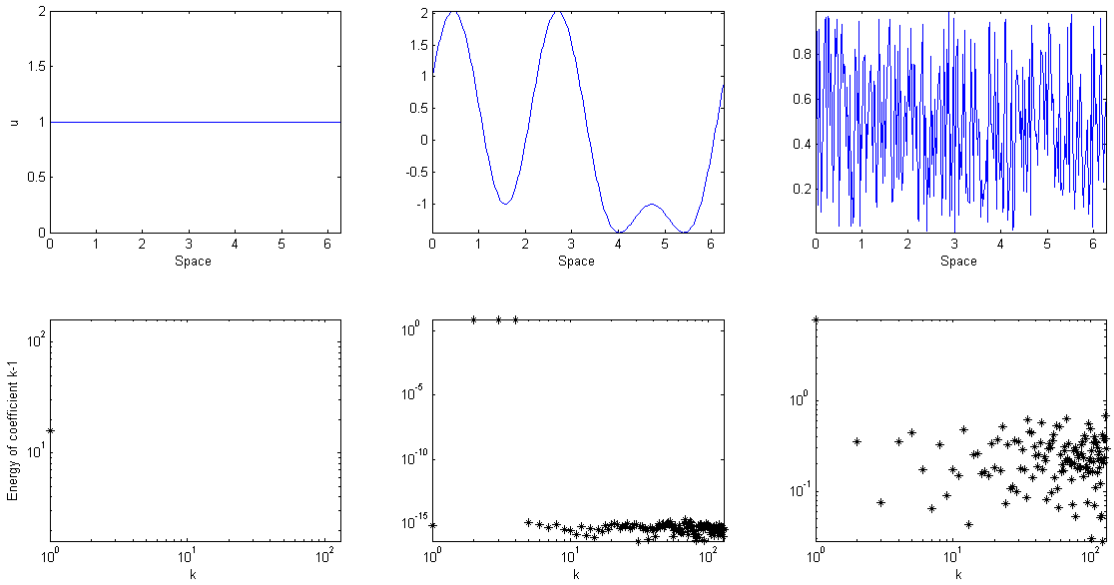

Figure 3.4: Three functions in real space (top) and in their power spectrums (bottom). Note that k on the x-axis for the power spectrums is associated with $\hat{u}_{k-1}$ in order to include the 0th wave number on the logarithmic axes. One notices that the more nonsmooth the functions are, the higher frequency they are, and subsequently the more energy is in their power spectrum

frequency functions u with their power spectrums.

Another thing to notice is that since our function will take on real values then we require conjugate symmetry of the coefficients, $\hat{u}_k = \overline{\hat{u}_{-k}}$. This is simple to notice, take any wave number, say k, of the sum of Fourier basis and coefficients, $\hat{u}_k\phi_k(x)$, and its opposite wave number, $-k$. If we split the real and complex components of their sum we would have

$$\begin{aligned}\hat{u}_k\phi_k(x) + \hat{u}_{-k}\phi_{-k}(x) &= (\hat{r}_k + i\hat{c}_k)(\cos(kx) + i\sin(kx)) + \\ &\quad (\hat{r}_{-k} + i\hat{c}_{-k})(\cos(kx) - i\sin(kx)), \\ &= ((\hat{r}_k + \hat{r}_{-k})\cos(kx) - (\hat{c}_k - \hat{c}_{-k})\sin(kx)) + \\ &\quad i((\hat{r}_k - \hat{r}_{-k})\sin(kx) + (\hat{c}_k + \hat{c}_{-k})\cos(kx))\end{aligned}$$

and if we have real valued u then we need the imaginary part of this sum to be zero.

This then forces

$$\hat{r}_k = \hat{r}_{-k}, \quad \hat{c}_k = -\hat{c}_{-k},$$

or simply

$$\hat{u}_k = \overline{\hat{u}_{-k}}.$$

Because of this we do not need to store roughly half the coefficients since we can derive half of them based on the other half. Therefore in practice we store only the coefficients of the positive wave numbers. Additionally we wish to separate the real and complex parts of the coefficients and store those by themselves in order to improve accuracy, eliminating numerical error of the real part computations from entering in the imaginary part computations and vice-versa. There is two other modifications we make in practice. Since we have the conjugate symmetry of coefficients then we have for the zeroth wave number that,

$$\hat{u}_0 = \overline{\hat{u}_{-0}},$$

$$\hat{r}_0 + i\hat{c}_0 = \hat{r}_0 - i\hat{c}_0$$

and therefore $\hat{c}_0 = 0$ so we need not store it. Finally, since

$$\phi_{\frac{N}{2}}(x_j) = \exp\left(i\frac{N}{2}\frac{2\pi}{N}j\right) = (-1)^j,$$

which is real valued then,

$$\hat{u}_{\frac{N}{2}}\phi_{\frac{N}{2}}(x_j) = (\hat{r}_{\frac{N}{2}} + i\hat{c}_{\frac{N}{2}})(-1)^j,$$

and we require this to be real valued as well so

$$\hat{c}_{\frac{N}{2}} = 0.$$

Therefore in practice we will store the real component of wave numbers $0, \ldots, \frac{N}{2}$ and the complex component of wave numbers $1, \ldots, \frac{N}{2} - 1$ and if $\hat{c}_0$ or $\hat{c}_{\frac{N}{2}}$ is ever referred to we set it zero.

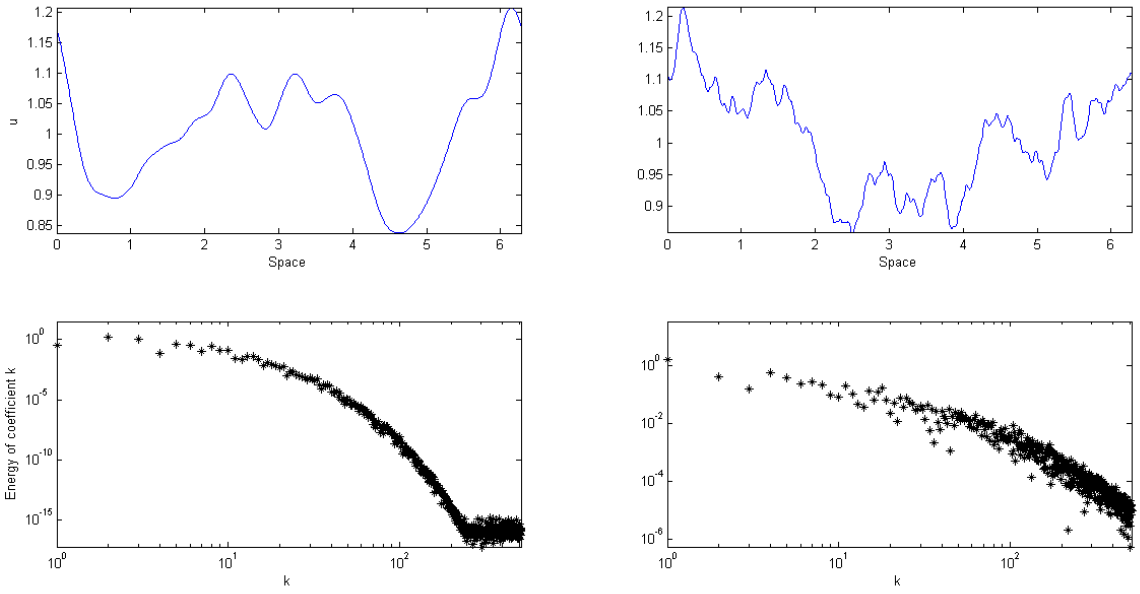

Figure 3.5: A well resolved solution (top left) and its power spectrum (bottom left) compared with a poorly resolved solution (top right) and its power spectrum (bottom right).

3.3 Convergence and advantages of pseudo-spectral methods

Now we need to have an idea of when our methods are working well and if the solutions are well resolved. The first question, whether the programs we create with pseudo-spectral methods are working properly, is typically answered by finite difference tests which we will cover in Chapter 4, Section 4.5. For the notion of a well resolved solution refer to Figure 3.5.

The power spectrum is what mainly tells us if a solution is well resolved. The power spectrum of a well resolved solution decays at least exponentially after some wave number and they decay to energies of at least 10^{-8}. In Figure 3.5 the energies of the well resolved solution decays to levels of numerical precision and it does so at least exponentially, which is the best possible resolution we can achieve. Conversely, the poorly resolved solution decays polynomially and only decays to energy levels of about 10^{-6}. What this means theoretically is that something is keeping the pseudo-spectral

methods from achieving the exponential decay and that there are wave numbers beyond our truncation which have non-negligible energies. Both of these would be issues we would have to address.

So in all our simulations, if the power spectrum does not decay exponentially or does not decay to low enough levels, then we should be suspicious of the accuracy of our results. If we do see these signs then we should try to increase the number of spatial grid points or look at other potential sources of error, specifically aliasing errors which will describe in Chapter 3, Section 3.4, Subsection 3.4.1.

As for the advantages of pseudo-spectral methods, we already described how they can allow for faster computational times when compared with finite difference time stepping. Another advantage is the ability to transform derivatives into scalar multiplication, which for equation (2.3.5), transforms our PDE to a system of ODEs. Specifically with our Fourier transform,

$$\widehat{(u_x)}_k = ik\hat{u}_k.$$

So if we take our system to Fourier space then we can do away with the spatial derivative and transform it to a system of ODEs in time with respect to the Fourier coefficients. Dealing with a system of ODEs in time is simpler than dealing with equation (2.3.5) and also because we are using periodic basis functions then we automatically satisfy the periodic boundary conditions.

The last advantage to these methods for our purposes comes into play with the computation of the interaction terms. The Convolution Theorem relates convolutions in real space with scalar multiplication in Fourier space and vice-versa. Specifically for our interaction terms,

$$\widehat{(u * v)}_k = \hat{u}_k\hat{v}_k,$$

giving us a simple and quick way to compute the convolutions. Instead of approximating the integrals or using other software to compute the convolutions we can compute the point-wise multiplication of the Fourier coefficients for u and v and transform back to real space and we will have our interaction terms.

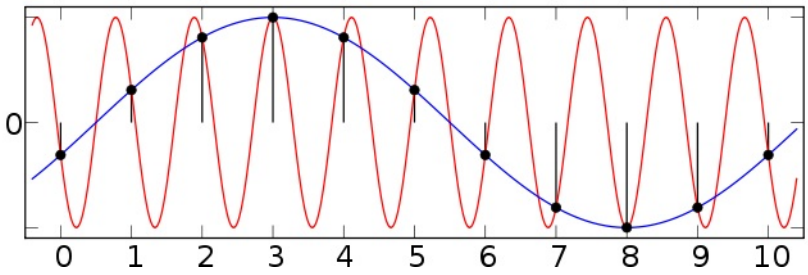

Figure 3.6: Two Fourier basis functions which are equal on every grid point. Taken from Trefethen [15].

3.4 Known sources of potential error

3.4.1 Aliasing errors

Aliasing errors arise naturally when using these methods on systems with nonlinearities and although it is known how to counteract these errors in some cases, it is not known how to counteract them in all cases. Aliasing errors come from discretizing space into a grid on which the basis functions which are N wave numbers apart are identical on the grid,

$$\phi_k(x_j) = \phi_{k+N}(x_j), \quad j = 0, \ldots, N-1.$$

Figure 3.6 shows an example of two basis functions which are equal on every grid point.

To understand aliasing errors better we will consider a simple, well-understood example. Consider the quadratic nonlinearity,

$$w = u^2,$$

and recall the Convolution Theorem from earlier. Assume u is given as shown in Figure 3.7, then formally we have that since u^2 is point-wise multiplication in real space, then the Fourier coefficients of w should be given by the convolution of the Fourier coefficients of u. Figure 3.8 shows w in real space and in Fourier space from this formal understanding. Notice we need twice the number of grid points to formally represent w, although it is only noticeable in the power spectrum.

Notice that w in Fourier space has energy in twice the wave numbers as wave

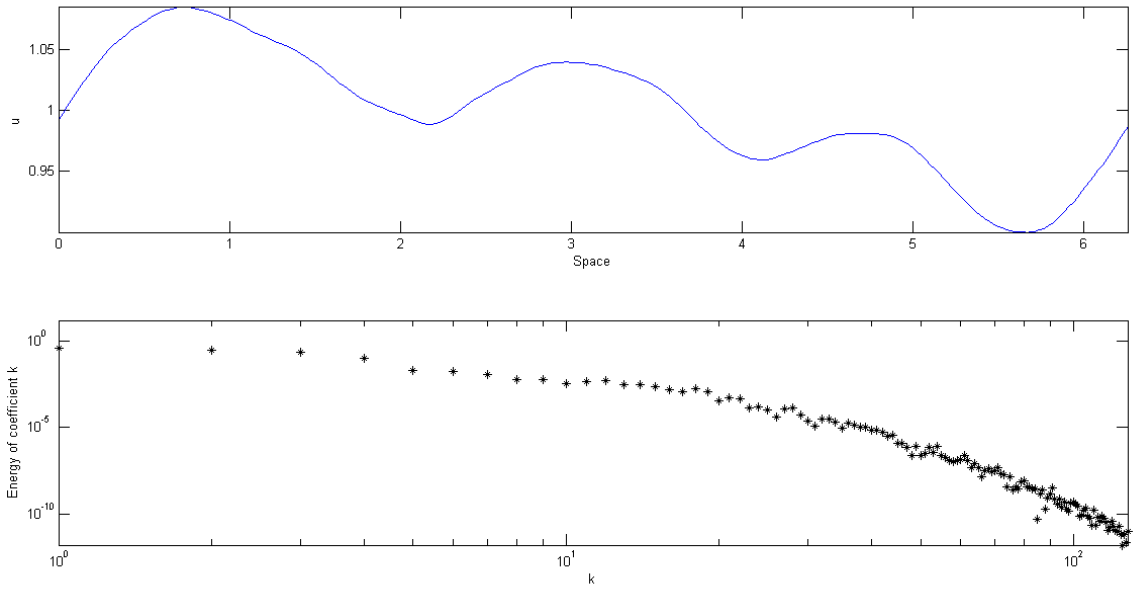

Figure 3.7: Real space representation of u (top) along with power spectrum (bottom).

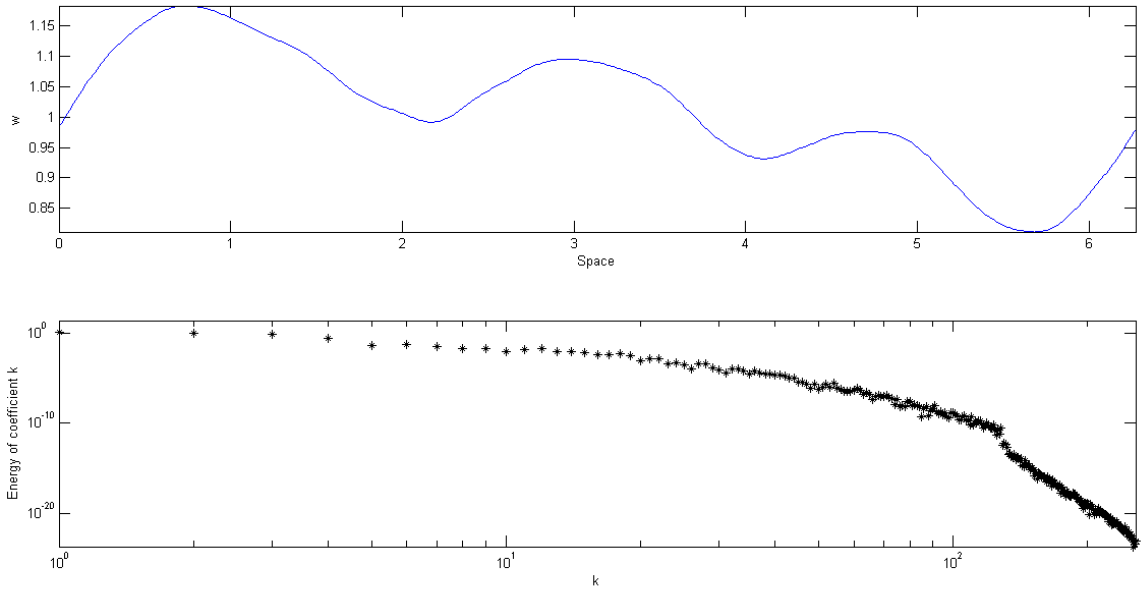

Figure 3.8: Real space representation of $w = u^2$ (top) along with power spectrum (bottom). Notice the quadratic nonlinearity needs twice the grid points to be properly resolved when compared to Figure 3.7.

numbers which had energy in the Fourier space representation of u. This is precisely where aliasing errors come from. In practice we would form u in real space, compute the point-wise multiplication,

$$w_j = u_j u_j, \quad j = 0, \ldots, N-1,$$

then take w to Fourier space. This would be using the truncation on our basis functions, ϕ_k, we had for u. Thus the energy from the formal definition of $\hat{w}_k$ in wave numbers $[-N, -\frac{N}{2}]$ alias to wave numbers $[0, \frac{N}{2}]$ and similarly, energy in wave numbers $(\frac{N}{2}, N]$ alias to wave numbers $(-\frac{N}{2}, 0]$.

So how can we first of all notice these errors coming about and secondly, how can we mitigate this error? The most prominent sign of aliasing errors comes from the power spectrum. As we mentioned earlier, we want a well resolved solution whose power spectrum decays exponentially. Aliasing errors will cause energy from the wave numbers past our truncation to move into the wave numbers of our truncation and is therefore error that should not be there.

Although if a nonlinearity exists in the equations then aliasing errors will always occur if we apply pseudo-spectral methods naively, in less severe cases it may not be noticeable at all. In more severe cases, however, the thing one notices most is that the power spectrum will not decay exponentially. This is the primary sign of aliasing errors, but one should expect them to exist in any case where pseudo-spectral methods are applied to nonlinearities without concern to these errors.

So now we have a grasp on what they are and how to notice them, how do we mitigate them? The idea is fairly simple and there are two variations that accomplish the same goal. The basic idea is to have zeros in all wave numbers that would cause aliasing errors. If we set zero all coefficients for wave numbers $k > \frac{N}{4}$ then our quadratic nonlinearity would formally have energy in wave numbers up to twice this cut off, namely $2\frac{N}{4} = \frac{N}{2}$, which is the truncation we have on u originally. So formally, no energy would be in wave numbers outside our truncation and so no aliasing errors.

One thing we must watch for when we do this zeroing out of coefficients is whether we are removing significant wave numbers. If we zero out coefficients with significant

energy, anything higher than 10^{-8}, then we could be introducing non-negligible error. If we are dealing with well resolved solutions then this perhaps will not be a problem. But, if we do notice we would zero out significant wave numbers, then we could merely increase the spatial resolution until this problem can be avoided.

We have discussed how to remove aliasing error from quadratic nonlinearities, and indeed the basic process works with any polynomial nonlinearity. Essentially for any polynomial nonlinearity, u^p, with $p \in \mathbb{Z}$ the energy diffuses to wave numbers up to p times the maximum wave number with energy of u. So if our truncation is up to $\frac{N}{2}$ then we could zero out coefficients past wave number $\frac{N}{2p}$ in a similar fashion to what we described and remove the aliasing error. We should mention that one can zero out less than this and still have the de-aliasing scheme work but refer the reader to Trefethen [15].

This only works well for polynomial nonlinearities. In our model we have a hyperbolic tangent term that might cause problems because of aliasing errors. If we expand it out as its Taylor series this perhaps implies that the energy diffuses up to any wave number. Figure 3.9 shows us a solution and its power spectrum from simulations where aliasing errors may be significantly affecting the simulation. Although the power spectrum does decay to numerical precision, it does not do so exponentially which is a sign of aliasing errors as we discussed.

3.4.2 Gibb's phenomenon

The Gibb's phenomenon is another potential source of error for our model and has to do with the steepness of gradients along with the spatial discretization. Theoretically the Gibb's phenomenon occurs when you try to write a discontinuous function as a sum of Fourier basis functions. No matter the truncation you choose on the basis functions, ripples centered on the discontinuity will appear. As you increase the truncation on your Fourier approximation one notices that the ripples become more localized to the discontinuity but will maintain a finite amplitude regardless of how high of a truncation you take. Figure 3.10 shows this quite clearly for various truncations on the Fourier basis. Notice the ripples persist no matter how high a truncation we take.

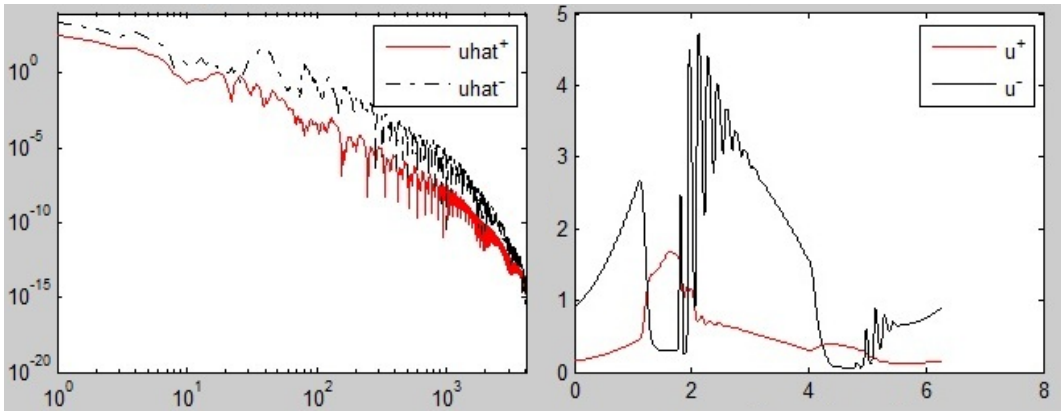

Figure 3.9: Power spectrum of solution (left) along with the final time plot of density distributions (right) for a simulation is showing signs of aliasing errors as can be seen by the polynomial decay of the power spectrum. A Gibb's phenomenon type error is also showing signs as ripples are evident on density distributions.

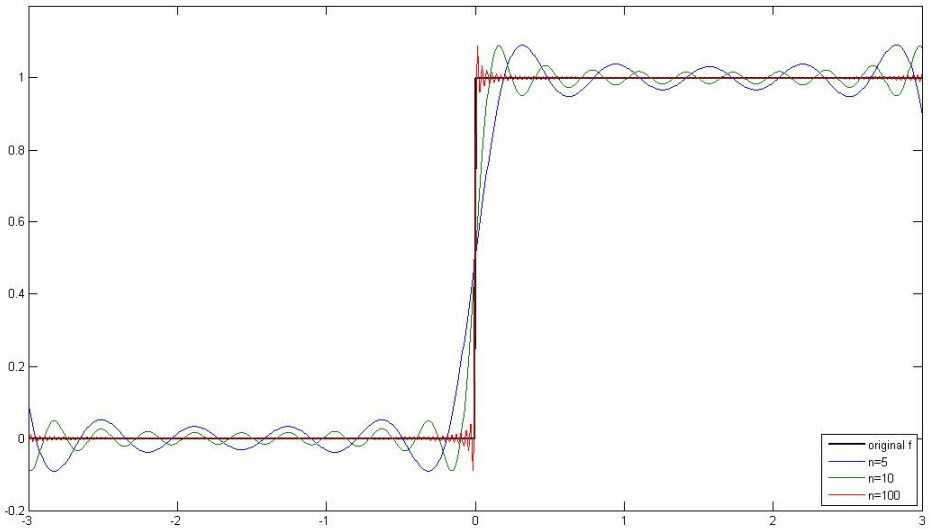

Figure 3.10: Fourier approximations truncated to the nth wave number compared with the true discontinuity. Notice the ripples retain a finite amplitude but become more localized to the discontinuity. Taken from http://www.charlesgao.com/en/?p=136 July 3rd, 2013.

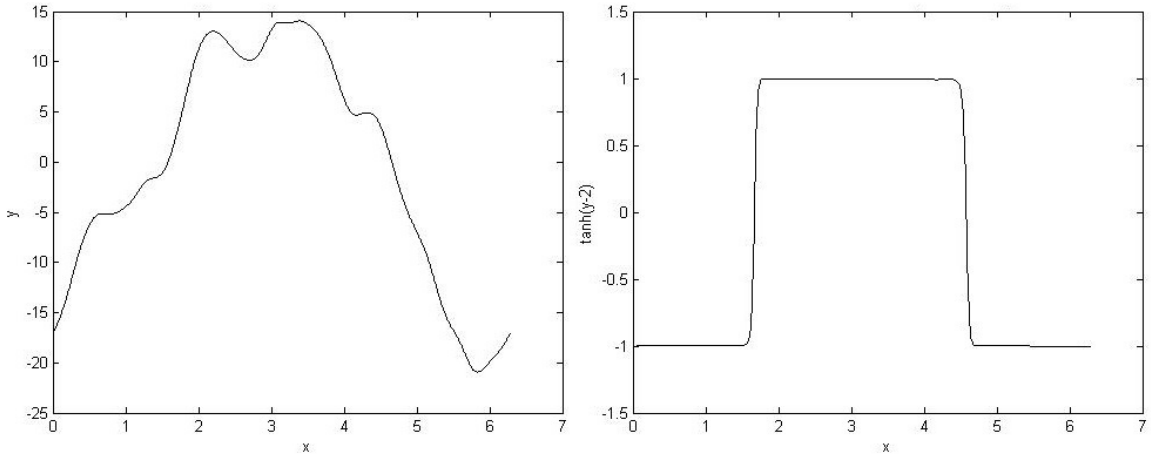

Figure 3.11: Comparison between a total interaction term, y, from simulations (left), and the same term passed through the shifted hyperbolic tangent, $\tanh(y-y_0)$ (right). The steep gradients of $\tanh(y-y_0)$ could cause Gibb's phenomenon type errors.

The Gibb's phenomenon can occur even if it is not a true discontinuity. If the solution has a steep gradient and/or our spatial grid is coarse enough then the steep gradient might look like a discontinuity on the grid. If this happens then we could get the same kind of ripples occurring centered on the steep gradient. Figure 3.3 already showcased a solution with a particularly steep gradient and as mentioned we do not know the extent to how nonsmooth solutions may be.

Again the hyperbolic tangent term could be a cause of this error. If the total interaction term, y, has large amplitude components then once it is passed through the hyperbolic tangent the result may appear to be almost like a square wave as Figure 3.11 depicts.

If our grid does not resolve these steep gradients arising from the hyperbolic tangent then we may indeed see Gibb's phenomenon affecting simulations. In fact we do see simulations where this may be happening as highlighted in Figure 3.9. The drastic ripples on the solution is a tell-tale sign of Gibb's phenomenon, although it is not absolute proof.

Chapter 4

Time stepping the system and the first variational equation

4.1 Applying the Fourier transform to the PDE

So now we need to apply the Fourier transform to equation (2.3.5) to get a system of ODEs in time with respect to the Fourier coefficients. We will let,

$$NLT = u^+ \tanh(y - y_0) - u^- \tanh(-y - y_0),$$

so we may write equation (2.3.5) as

$$\mp\partial_t u^\pm = \gamma\partial_x u^\pm + \Lambda(u^+ - u^-) + \frac{1}{2}\lambda_2 NLT.$$

Now apply our spatial grid with the function values of $u^\pm$ and NLT on the grid to get

$$\mp\partial_t u_j^\pm = \gamma\partial_x u_j^\pm + \Lambda(u_j^+ - u_j^-) + \frac{1}{2}\lambda_2 NLT_j, \quad j = 0, \ldots, N-1. \tag{4.1.1}$$

Now we apply our IDFT so we get

$$u_j^\pm = \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \hat{u}_k^\pm \phi_k(x_j), \qquad \partial_t u_j^\pm = \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \partial_t \hat{u}_k^\pm \phi_k(x_j),$$

$$\partial_x u_j^\pm = \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} (ik\hat{u}_k^\pm) \phi_k(x_j), \qquad NLT_j = \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \widehat{NLT}_k \phi_k(x_j),$$

and we get equation (4.1.1) transformed as

$$\sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \left(\mp\partial_t \hat{u}_k^\pm - ik\gamma\hat{u}_k^\pm - \Lambda(\hat{u}_k^+ - \hat{u}_k^-) - \frac{1}{2}\lambda_2 \widehat{NLT}_k \right) \phi_k(x_j) = 0, \tag{4.1.2}$$

for $j = 0, \ldots, N-1$. We then introduce the inner product,

$$\langle u, v \rangle = \sum_{j=0}^{N-1} u_j \bar{v}_j, \quad u, v \in \mathbb{C}^N.$$

Note that $\phi_k(x_j)$ and $\phi_q(x_j)$ are orthogonal if $k \neq q$ and so $\langle \phi_k(x_j), \phi_q(x_j) \rangle = N\delta_{k,q}$. At this point we take the condition that the left-hand side of equation (4.1.2) be orthogonal to the Fourier basis functions,

$$\langle \phi_q, \text{LHS} \rangle = 0, \quad q = -\frac{N}{2} + 1, \ldots, \frac{N}{2},$$

and get

$$\sum_{j=0}^{N-1} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \overline{\left(\mp \partial_t \hat{u}_k^{\pm} - ik\gamma \hat{u}_k^{\pm} - \Lambda(\hat{u}_k^{+} - \hat{u}_k^{-}) - \frac{1}{2}\lambda_2 \widehat{NLT}_k \right)} \phi_q(x_j)\phi_k(x_j)^{-1} = 0,$$

$$\sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \overline{\left(\mp \partial_t \hat{u}_k^{\pm} - ik\gamma \hat{u}_k^{\pm} - \Lambda(\hat{u}_k^{+} - \hat{u}_k^{-}) - \frac{1}{2}\lambda_2 \widehat{NLT}_k \right)} \sum_{j=0}^{N-1} (\phi_q(x_j)\phi_k(x_j)^{-1}) = 0,$$

$$\sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \overline{\left(\mp \partial_t \hat{u}_k^{\pm} - ik\gamma \hat{u}_k^{\pm} - \Lambda(\hat{u}_k^{+} - \hat{u}_k^{-}) - \frac{1}{2}\lambda_2 \widehat{NLT}_k \right)} \langle \phi_q(x_j), \phi_k(x_j) \rangle = 0,$$

$$\sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \overline{\left(\mp \partial_t \hat{u}_k^{\pm} - ik\gamma \hat{u}_k^{\pm} - \Lambda(\hat{u}_k^{+} - \hat{u}_k^{-}) - \frac{1}{2}\lambda_2 \widehat{NLT}_k \right)} N\delta_{q,k} = 0,$$

$$\overline{\left(\mp \partial_t \hat{u}_q^{\pm} - iq\gamma \hat{u}_q^{\pm} - \Lambda(\hat{u}_q^{+} - \hat{u}_q^{-}) - \frac{1}{2}\lambda_2 \widehat{NLT}_q \right)} = 0,$$

for $q = -\frac{N}{2}+1, \ldots, \frac{N}{2}$. At this point we will return to our k notation for the equations. Continuing along we get a system of ODEs as

$$\mp \partial_t \hat{u}_k^{\pm} = ik\gamma \hat{u}_k^{\pm} + \Lambda(\hat{u}_k^{+} - \hat{u}_k^{-}) + \frac{1}{2}\lambda_2 \widehat{NLT}_k, \quad k = -\frac{N}{2} + 1, \ldots, \frac{N}{2}. \tag{4.1.3}$$

As we mentioned in Chapter 3, Section 3.2, the strictly negative wave numbers are redundant to keep track of, so we can reduce this system by focusing just on $k = 0, \ldots, \frac{N}{2}$. Next we want to split the real and complex components of all terms. Thus we let

$$\hat{u}_k^{\pm} = \hat{r}_k^{\pm} + i\hat{c}_k^{\pm}, \quad \widehat{NLT}_k = \widehat{rNLT}_k + i\widehat{cNLT}_k,$$

so equation (4.1.3) becomes

$$\begin{aligned} \mp \partial_t \left(\hat{r}_k^{\pm} + i\hat{c}_k^{\pm} \right) =& ik\gamma \left(\hat{r}_k^{\pm} + i\hat{c}_k^{\pm} \right) + \Lambda \left(\left(\hat{r}_k^{+} + i\hat{c}_k^{+} \right) - \left(\hat{r}_k^{-} + i\hat{c}_k^{-} \right) \right) + \\ & \frac{1}{2}\lambda_2 \left(\widehat{rNLT}_k + i\widehat{cNLT}_k \right), \quad k = 0, \ldots, \frac{N}{2}. \end{aligned}$$

So splitting the real and complex components of this equation we get

$$\begin{aligned} \text{Real:} \quad \mp\partial_t \hat{r}_k^{\pm} &= -k\gamma \hat{c}_k^{\pm} + \Lambda(\hat{r}_k^{+} - \hat{r}_k^{-}) + \frac{1}{2}\lambda_2 \widehat{rNLT}_k, \quad k = 0, \ldots, \frac{N}{2}, \\ \text{Imag:} \quad \mp\partial_t \hat{c}_k^{\pm} &= k\gamma \hat{r}_k^{\pm} + \Lambda(\hat{c}_k^{+} - \hat{c}_k^{-}) + \frac{1}{2}\lambda_2 \widehat{cNLT}_k, \quad k = 1, \ldots, \frac{N}{2} - 1, \end{aligned} \tag{4.1.4}$$

noting that we are using the fact that we know $\hat{c}_0^{\pm} = 0$ and $\hat{c}_{\frac{N}{2}}^{\pm} = 0$ from Chapter 3, Section 3.2. Now we write equation (4.1.4) as the full 4 ODEs grouping together terms. Thus we get,

$$\begin{aligned} -\partial_t \hat{r}_k^{+} &= \Lambda \hat{r}_k^{+} - k\gamma \hat{c}_k^{+} - \Lambda \hat{r}_k^{-} + \frac{1}{2}\lambda_2 \widehat{rNLT}_k, \qquad k = 0, \ldots, \frac{N}{2}, \\ -\partial_t \hat{c}_k^{+} &= k\gamma \hat{r}_k^{+} + \Lambda \hat{c}_k^{+} - \Lambda \hat{c}_k^{-} + \frac{1}{2}\lambda_2 \widehat{cNLT}_k, \qquad k = 1, \ldots, \frac{N}{2} - 1, \\ \partial_t \hat{r}_k^{-} &= \Lambda \hat{r}_k^{+} - \Lambda \hat{r}_k^{-} - k\gamma \hat{c}_k^{-} + \frac{1}{2}\lambda_2 \widehat{rNLT}_k, \qquad k = 0, \ldots, \frac{N}{2}, \\ \partial_t \hat{c}_k^{-} &= \Lambda \hat{c}_k^{+} + k\gamma \hat{r}_k^{-} - \Lambda \hat{c}_k^{-} + \frac{1}{2}\lambda_2 \widehat{cNLT}_k, \qquad k = 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{4.1.5}$$

Now let

$$U = \left((\hat{r}^{+})^T, (\hat{c}^{+})^T, (\hat{r}^{-})^T, (\hat{c}^{-})^T\right)^T$$

where

$$\hat{r}^{\pm} = \left(\hat{r}_0^{\pm}, \ldots, \hat{r}_{\frac{N}{2}}^{\pm}\right)^T, \quad \hat{c}^{\pm} = \left(\hat{c}_1^{\pm}, \ldots, \hat{c}_{\frac{N}{2}-1}^{\pm}\right)^T.$$

Then let

$$\Omega = \begin{pmatrix} -I_{\frac{N}{2}+1} & 0 & 0 & 0 \\ 0 & -I_{\frac{N}{2}-1} & 0 & 0 \\ 0 & 0 & I_{\frac{N}{2}+1} & 0 \\ 0 & 0 & 0 & I_{\frac{N}{2}-1} \end{pmatrix},$$

let K be a $\frac{N}{2}+1$ by $\frac{N}{2}-1$ matrix such that $K_{k+1,k} = k\gamma$ for $k = 1, \ldots, \frac{N}{2}-1$ and $K_{q,k} = 0$ otherwise. Finally let,

$$Q = \begin{pmatrix} \Lambda I_{\frac{N}{2}+1} & -K & -\Lambda I_{\frac{N}{2}+1} & 0 \\ K^T & \Lambda I_{\frac{N}{2}-1} & 0 & -\Lambda I_{\frac{N}{2}-1} \\ \Lambda I_{\frac{N}{2}+1} & 0 & -\Lambda I_{\frac{N}{2}+1} & -K \\ 0 & \Lambda I_{\frac{N}{2}-1} & K^T & -\Lambda I_{\frac{N}{2}-1} \end{pmatrix},$$

so we may write the entire system of ODEs for our Fourier coefficients as,

$$\Omega \partial_t U = QU + NLTv \tag{4.1.6}$$

where

$$NLTv = \frac{1}{2}\lambda_2 \left(\widehat{rNLT}^T, \widehat{cNLT}^T, \widehat{rNLT}^T, \widehat{cNLT}^T\right)^T$$

and

$$\widehat{rNLT} = \left(\widehat{rNLT}_0, \ldots, \widehat{rNLT}_{\frac{N}{2}}\right)^T, \quad \widehat{cNLT} = \left(\widehat{cNLT}_1, \ldots, \widehat{cNLT}_{\frac{N}{2}-1}\right)^T.$$

4.2 Computation of nonlinear terms

Now we have a formulation for the entire system of ODEs in time with respect to the Fourier coefficients. To make it complete though we need a way to compute the nonlinear terms and transfer to Fourier space. Thus we restate our nonlinear terms,

$$NLT = u^+ \tanh(y - y_0) - u^- \tanh(-y - y_0), \quad y = \sum_{i=\{a,r,al\}} q_i \left(K_i \star u^- - K_i * u^+\right).$$

We will begin with the most internal terms, namely $K_i \star u^-$ and $K_i * u^+$. For the first we have

$$K_i \star u^- = \int_{-\infty}^{\infty} K_i(s) u^-(x+s) ds. \tag{4.2.1}$$

In order to apply the Convolution Theorem we need to change this cross-correlation into a proper convolution. Thus let us do a change of variables such that,

$$s = -r, \quad ds = -dr,$$

and call

$$Q_i(r) = K_i(-r)$$

so we may write the equation (4.2.1) as

$$-\int_{\infty}^{-\infty} Q_i(r) u^-(x-r) dr$$

and then switch the bounds of integration so we write equation (4.2.1) as a convolution,

$$\int_{-\infty}^{\infty} Q_i(r) u^-(x-r) dr = Q_i * u^-.$$

Now we use the Convolution Theorem and thus know,

$$\widehat{Q_i * u^-}_k = \widehat{(Q_i)}_k \hat{u}_k^-,$$
$$\widehat{K_i * u^+}_k = \widehat{(K_i)}_k \hat{u}_k^+,$$

and we have the Fourier transform of Q_i and K_i can be found formally as,

$$\widehat{(Q_i)}_k = \exp\left(-\frac{1}{128}k^2 s_i^2 + iks_i\right),$$
$$\widehat{(K_i)}_k = \exp\left(-\frac{1}{128}k^2 s_i^2 - iks_i\right).$$

Therefore we have

$$\begin{aligned}\widehat{(Q_i)}_k \hat{u}_k^- &= \exp\left(-\frac{1}{128}k^2 s_i^2\right)(\cos(ks_i) + i\sin(ks_i))\left(\hat{r}_k^- + i\hat{c}_k^-\right), \\ &= \exp\left(-\frac{1}{128}k^2 s_i^2\right)\left((\cos(ks_i)\hat{r}_k^- - \sin(ks_i)\hat{c}_k^-) + i(\cos(ks_i)\hat{c}_k^- + \sin(ks_i)\hat{r}_k^-)\right)\end{aligned}$$

and

$$\begin{aligned}\widehat{(K_i)}_k \hat{u}_k^+ &= \exp\left(-\frac{1}{128}k^2 s_i^2\right)(\cos(ks_i) - i\sin(ks_i))\left(\hat{r}_k^+ + i\hat{c}_k^+\right), \\ &= \exp\left(-\frac{1}{128}k^2 s_i^2\right)\left((\cos(ks_i)\hat{r}_k^+ + \sin(ks_i)\hat{c}_k^+) + i(\cos(ks_i)\hat{c}_k^+ - \sin(ks_i)\hat{r}_k^+)\right).\end{aligned}$$

Putting these together we have

$$\begin{aligned}\hat{y}_{ik} =& q_i(\exp\left(-\frac{1}{128}k^2 s_i^2\right)\left((\cos(ks_i)\hat{r}_k^- - \sin(ks_i)\hat{c}_k^-) + i(\cos(ks_i)\hat{c}_k^- + \sin(ks_i)\hat{r}_k^-)\right) - \\ & \exp\left(-\frac{1}{128}k^2 s_i^2\right)\left((\cos(ks_i)\hat{r}_k^+ + \sin(ks_i)\hat{c}_k^+) + i(\cos(ks_i)\hat{c}_k^+ - \sin(ks_i)\hat{r}_k^+)\right)). \\ =& q_i \exp\left(-\frac{1}{128}k^2 s_i^2\right)\left(\left(\cos(ks_i)(\hat{r}_k^- - \hat{r}_k^+) - \sin(ks_i)(\hat{c}_k^+ + \hat{c}_k^-)\right) + \right. \\ & \left. i\left(\cos(ks_i)(\hat{c}_k^- - \hat{c}_k^+) + \sin(ks_i)(\hat{r}_k^- + \hat{r}_k^+)\right)\right).\end{aligned}$$

Furthermore if we let

$$C_{k,i} = q_i \exp\left(-\frac{1}{128}k^2 s_i^2\right)\cos(ks_i), \quad S_{k,i} = q_i \exp\left(-\frac{1}{128}k^2 s_i^2\right)\sin(ks_i),$$

then write

$$C_k = \sum_{i=\{a,r,al\}} C_{k,i}, \quad S_k = \sum_{i=\{a,r,al\}} S_{k,i}.$$

With these we can write our total interaction term succinctly in Fourier space with real and complex splitting as

$$\Re(\hat{y}_k) = C_k(\hat{r}_k^- - \hat{r}_k^+) - S_k(\hat{c}_k^+ + \hat{c}_k^-),$$
$$\Im(\hat{y}_k) = C_k(\hat{c}_k^- - \hat{c}_k^+) + S_k(\hat{r}_k^- + \hat{r}_k^+).$$

And once we have these terms in Fourier space then we take $\hat{y}$ into real space with our IDFT to get y. We can already take $\hat{u}^\pm$ into real space and therefore we can compute

$$NLT = u^+ \tanh(y - y_0) - u^- \tanh(-y - y_0).$$

If we were attempting to de-alias this nonlinearity we would pad $\hat{u}^\pm$ and $\hat{y}$ with zeros before taking them to real space to compute NLT. However we have not attempted to de-alias this nonlinearity as it is not well known how exactly the hyperbolic tangent causes aliasing errors. After we have NLT we take it to Fourier space with our DFT and then we have our nonlinear terms in Fourier space. Then our application of pseudo-spectral methods to the PDE is complete and we are ready to discretize temporally and start time-stepping.

4.3 Temporal discretization and initial condition of time-stepping

At this point we need to time-step equation (4.1.6). To begin, we form our temporal grid as

$$t_m = \Delta m, \quad m = 0, \ldots, M,$$

such that we wish to evolve an initial condition to time ΔM. Next we need to choose our function values on our temporal grid. For the nonlinear terms we will let,

$$U(t_m) \approx U_m,$$

and thus,

$$NLTv(U(t_m)) \approx NLTv(U_m) = NLTv_m,$$

so we have an explicit scheme for the nonlinear terms when attempting to time-step to t_{m+1} from t_m. For the linear terms we will let

$$U(t_m) \approx \frac{U_m + U_{m+1}}{2}, \quad \partial_t U(t_m) \approx \frac{U_{m+1} - U_m}{\Delta},$$

such that we use a trapezoidal method for the U term and a forward Euler method for the $\partial_t U$ term. Thus we have a semi-implicit scheme for the linear terms and an explicit scheme for the nonlinear terms. This choice was originally motivated by the transport equation portion of the equations, knowing that a trapezoidal method preserves amplitudes when time-stepping the transport equation. One can refer to Ascher and Petzold for a more in-depth discussion of stability and convergence of the trapezoidal method [1].

Simulations do show that this choice of discretization is working well as we will show in Chapter 4, Section 4.5 with comparison to computed solutions from simulations done by Eftimie *et al.* [5], as well as finite difference tests. Though perhaps there is a more suitable discretization scheme.

With this discretization choice equation (4.1.6) separates into a system to solve for U_{m+1} as,

$$\begin{aligned}
\Omega\left(\frac{U_{m+1}-U_m}{\Delta}\right) &= Q\left(\frac{U_{m+1}+U_m}{2}\right)+\frac{1}{2}\lambda_2 NLTv_m,\\
2\Omega(U_{m+1}-U_m) &= \Delta Q(U_{m+1}+U_m)+\Delta\lambda_2 NLTv_m,\\
(2\Omega-\Delta Q)U_{m+1} &= (2\Omega+\Delta Q)U_m+\Delta\lambda_2 NLTv_m.
\end{aligned}$$

Thus if we wish to time-step a solution U_m to the next time-step U_{m+1} we must solve the system

$$(2\Omega-\Delta Q)U_{m+1} = (2\Omega+\Delta Q)U_m+\Delta\lambda_2 NLTv_m. \tag{4.3.1}$$

In practice we take our known solution, be it an initial condition or an already time-stepped solution, compute the nonlinear terms as outlined in the previous section, and then solve equation (4.3.1) for the solution at the next time-step.

Next we address what will be our initial condition in practice. We mentioned in Section 2 that we will use the homogeneous steady state with $A^* = 2$. Namely we will use small perturbations from

$$(u^+, u^-) = (1, 1).$$

We will take these perturbations in the same way as done by Eftimie *et al.* [5], which is to take 0.01 amplitude random noise and add it to the homogeneous steady state.

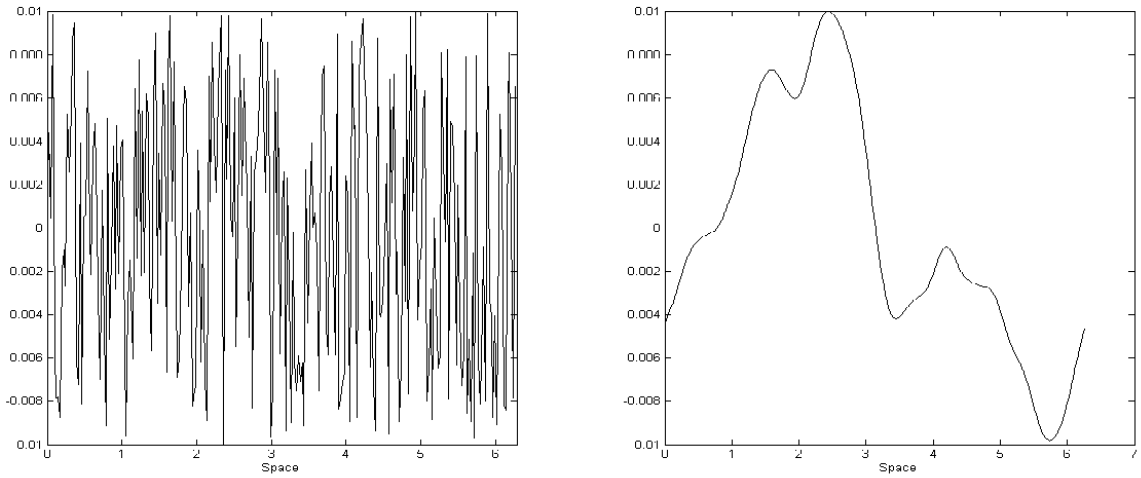

Figure 4.1: Unfiltered random noise (left) compared with filtered random noise (right). Filter is $\exp\left(-k^{\frac{2}{3}}\right)$ with post-processing in order to retain amplitude and mean.

If we call X_N the space of N by 1 vectors whose entries are randomly chosen between -0.01 and 0.01 then our initial conditions for simulations will be precisely,

$$(u_0^+, u_0^-)^T = (1_{1,N}, 1_{1,N})^T + (v^T, z^T)^T, \quad v, z \in X_N.$$

So we will set this initial condition initially, transform it to Fourier space, and use this result as U_0. From there we can time-step the equations. We note that in practice we smooth out the random noise with a filter, $\exp\left(-k^{\frac{2}{3}}\right)$, with post-processing so that the filtered noise retains the same amplitude and mean. This is important for initial conditions as the unfiltered noise would have a poorly resolved power spectrum. Figure 4.1 displays the unfiltered random noise and the filtered random noise.

4.4 The first variational equation

4.4.1 Explanation and derivation

Now we have a way to evolve a given state of right- and left-moving populations according to equation (2.3.5). However, in order to use the continuation methods we will introduce in Section 5 we will need a way to evolve a given state, (u^+, u^-), as well as perturbations with respect to the state, call these (w^+, w^-). We will also need

to evolve parameters, q_i, and perturbations to these parameters, say dq_i, although the equations describing their evolution will be trivial as we will show. The first variational equation is precisely the system to evolve all the information we will need for the continuation method.

We start by restating our PDE as

$$\partial_t u = f(u, q_\ell)$$

where we recall the definitions of u and f from equation (2.3.6). We note that although f does depend on each q_i for $i \in \{a, r, al\}$, in practice we only vary one of these parameters in the continuation method so we need only denote how f depends on the parameter we vary which we will call q_ℓ. We also add an equation for the evolution of the parameter which is,

$$\partial_t q_\ell = 0,$$

so now the system

$$\begin{aligned} \partial_t u &= f(u, q_\ell), \\ \partial_t q_\ell &= 0, \end{aligned} \tag{4.4.1}$$

describes the evolution of our state and parameter. Next we need a system to describe the evolution of the perturbations to the state and parameter. Let $||\cdot||_{L_2}$ represent the L2-norm. For $\epsilon > 0$, we add some perturbation $w = (w^+, w^-)^T$ and dq_ℓ to our state and parameter such that $||w||_{L_2} = 1$ and $dq_\ell = 1$. We note that if $\ell = a$ then we set

$$dq_\ell \to -dq_\ell$$

since we did the same to q_a in Chapter 2, Section 2.3 and will aid us in simplifying terms later. Therefore we set

$$u \to u + \epsilon w, \quad q_\ell \to q_\ell + \epsilon dq_\ell$$

and then use a Taylor expansion of f about the state and parameter to transform

equation (4.4.1) into

$$\begin{aligned}\partial_t u + \epsilon \partial_t w &= f(u, q_\ell) + \epsilon Df(u, q_\ell)(w, dq_\ell)^T,\\ \partial_t q_\ell + \epsilon \partial_t dq_\ell &= 0,\end{aligned}$$

but some of the terms of this system are redundant as we will be evolving the state and parameter according to equation (4.4.1) simultaneously so in fact we can cancel terms and drop the ϵ to get

$$\begin{aligned}\partial_t w &= Df(u, q_\ell)(w, dq_\ell)^T,\\ \partial_t dq_\ell &= 0.\end{aligned} \tag{4.4.2}$$

Equation (4.4.2) is our first variational system describing the evolution of the perturbations. Now we need to compute Df and its action on $(w, dq_\ell)^T$. In order to accomplish this, we distinguish the components of f as

$$f(u, q_\ell) = \begin{pmatrix} f^{(1)}(u, q_\ell) \\ f^{(2)}(u, q_\ell) \end{pmatrix}$$

such that

$$f^{(1)}(u, q_\ell) = -\gamma \partial_x u^+ - \Lambda(u^+ - u^-) - \frac{1}{2}\lambda_2(u^+ \tanh(y - y_0) - u^- \tanh(-y - y_0))$$

and

$$f^{(2)}(u, q_\ell) = \gamma \partial_x u^- + \Lambda(u^+ - u^-) + \frac{1}{2}\lambda_2(u^+ \tanh(y - y_0) - u^- \tanh(-y - y_0)).$$

Therefore we have

$$Df(u, q_\ell) = \begin{pmatrix} f^{(1)}_{u^+} & f^{(1)}_{u^-} & f^{(1)}_{q_\ell} \\ f^{(2)}_{u^+} & f^{(2)}_{u^-} & f^{(2)}_{q_\ell} \end{pmatrix}. \tag{4.4.3}$$

We proceed with the derivations of the partial derivatives we need. For $f^{(1)}$ we get,

$$\begin{aligned} f^{(1)}_{u^+} &= -\gamma \partial_x(\,\cdot\,) - \Lambda - \frac{1}{2}\lambda_2 \partial_{u^+}(u^+ \tanh(y - y_0) - u^- \tanh(-y - y_0)),\\ f^{(1)}_{u^-} &= \Lambda - \frac{1}{2}\lambda_2 \partial_{u^-}(u^+ \tanh(y - y_0) - u^- \tanh(-y - y_0)),\\ f^{(1)}_{q_\ell} &= -\frac{1}{2}\lambda_2 \partial_{q_\ell}(u^+ \tanh(y - y_0) - u^- \tanh(-y - y_0)).\end{aligned}$$

Furthermore we have

$$\partial_{u^+}(u^+\tanh(y-y_0)-u^-\tanh(-y-y_0)) = \tanh(y-y_0)+(u^+\mathrm{sech}^2(y-y_0)+ u^-\mathrm{sech}^2(-y-y_0))\partial_{u^+}y,$$

$$\partial_{u^-}(u^+\tanh(y-y_0)-u^-\tanh(-y-y_0)) = -\tanh(-y-y_0)+(u^+\mathrm{sech}^2(y-y_0)+ u^-\mathrm{sech}^2(-y-y_0))\partial_{u^-}y,$$

$$\partial_{q_\ell}(u^+\tanh(y-y_0)-u^-\tanh(-y-y_0)) = (u^+\mathrm{sech}^2(y-y_0)+u^-\mathrm{sech}^2(-y-y_0))\partial_{q_\ell}y.$$

And finally we have

$$\partial_{u^+}y = \partial_{u^+}\sum_{i=\{a,r,al\}} q_i(K_i\star u^- - K_i * u^+) = -\sum_{i=\{a,r,al\}} q_iK_i*(\,\cdot\,).$$

$$\partial_{u^-}y = \partial_{u^-}\sum_{i=\{a,r,al\}} q_i(K_i\star u^- - K_i * u^+) = \sum_{i=\{a,r,al\}} q_iK_i\star(\,\cdot\,).$$

$$\partial_{q_\ell}y = \partial_{q_\ell}\sum_{i=\{a,r,al\}} q_i(K_i\star u^- - K_i * u^+) = (K_\ell\star u^- - K_\ell * u^+).$$

For brevity we will neglect to write the full partial derivatives and combine them in pieces later. We still need the partial derivatives of $f^{(2)}$, but if we were to go about the same process as above we see,

$$f^{(2)}_{u^+} = \Lambda + \frac{1}{2}\lambda_2\partial_{u^+}(u^+\tanh(y-y_0)-u^-\tanh(-y-y_0)),$$

$$f^{(2)}_{u^-} = \gamma\partial_x(\,\cdot\,) - \Lambda + \frac{1}{2}\lambda_2\partial_{u^-}(u^+\tanh(y-y_0)-u^-\tanh(-y-y_0)),$$

$$f^{(2)}_{q_\ell} = \frac{1}{2}\lambda_2\partial_{q_\ell}(u^+\tanh(y-y_0)-u^-\tanh(-y-y_0)).$$

So in fact these partial derivatives differ very slightly from those of $f^{(1)}$ and as such we have already derived the partial derivatives of the nonlinear terms of $f^{(2)}$. Now we wish to write out $Df(u,q_\ell)(w,dq_\ell)^T$ explicitly so we go through each term interaction

now.

$$
\begin{aligned}
f^{(1)}_{u^+}w^+ =& -\gamma\partial_x w^+ - \Lambda w^+ - \frac{1}{2}\lambda_2 w^+\tanh(y-y_0) - \frac{1}{2}\lambda_2(u^+\mathrm{sech}^2(y-y_0)+\\
& u^-\mathrm{sech}^2(-y-y_0))\left(-\sum_{i=\{a,r,al\}} q_i K_i * w^+\right),\\
f^{(1)}_{u^-}w^- =& \Lambda w^- + \frac{1}{2}\lambda_2 w^-\tanh(-y-y_0) - \frac{1}{2}\lambda_2(u^+\mathrm{sech}^2(y-y_0)+\\
& u^-\mathrm{sech}^2(-y-y_0))\left(\sum_{i=\{a,r,al\}} q_i K_i \star w^-\right),\\
f^{(1)}_{q_\ell}dq_\ell =& -\frac{1}{2}\lambda_2(u^+\mathrm{sech}^2(y-y_0)+u^-\mathrm{sech}^2(-y-y_0))\left(dq_\ell(K_\ell\star u^- - K_\ell * u^+)\right),\\
f^{(2)}_{u^+}w^+ =& \Lambda w^+ + \frac{1}{2}\lambda_2 w^+\tanh(y-y_0) + \frac{1}{2}\lambda_2(u^+\mathrm{sech}^2(y-y_0)+\\
& u^-\mathrm{sech}^2(-y-y_0))\left(-\sum_{i=\{a,r,al\}} q_i K_i * w^+\right),\\
f^{(2)}_{u^-}w^- =& \gamma\partial_x w^- - \Lambda w^- - \frac{1}{2}\lambda_2 w^-\tanh(-y-y_0) + \frac{1}{2}\lambda_2(u^+\mathrm{sech}^2(y-y_0)+\\
& u^-\mathrm{sech}^2(-y-y_0))\left(\sum_{i=\{a,r,al\}} q_i K_i \star w^-\right),\\
f^{(2)}_{q_\ell}dq_\ell =& \frac{1}{2}\lambda_2(u^+\mathrm{sech}^2(y-y_0)+u^-\mathrm{sech}^2(-y-y_0))\left(dq_\ell(K_\ell\star u^- - K_\ell * u^+)\right).
\end{aligned}
$$

Then we have

$$
\begin{aligned}
&{f^{(1)}}_{u^+}w^+ + f^{(1)}_{u^-}w^- + f^{(1)}_{q_\ell}dq_\ell = \\
&\quad -\gamma\partial_x w^+ - \Lambda(w^+ - w^-) - \frac{1}{2}\lambda_2(w^+\tanh(y-y_0) - w^-\tanh(-y-y_0)) - \\
&\quad \frac{1}{2}\lambda_2(u^+\mathrm{sech}^2(y-y_0) + u^-\mathrm{sech}^2(-y-y_0)) \\
&\quad \left(dq_\ell(K_\ell \star u^- - K_\ell * u^+) + \sum_{i=\{a,r,al\}} q_i(K_i \star w^- - K_i * w^+)\right) \\
&{f^{(2)}}_{u^+}w^+ + f^{(2)}_{u^-}w^- + f^{(2)}_{q_\ell}dq_\ell = \\
&\quad \gamma\partial_x w^- + \Lambda(w^+ - w^-) + \frac{1}{2}\lambda_2(w^+\tanh(y-y_0) - w^-\tanh(-y-y_0)) + \\
&\quad \frac{1}{2}\lambda_2(u^+\mathrm{sech}^2(y-y_0) + u^-\mathrm{sech}^2(-y-y_0)) \\
&\quad \left(dq_\ell(K_\ell \star u^- - K_\ell * u^+) + \sum_{i=\{a,r,al\}} q_i(K_i \star w^- - K_i * w^+)\right).
\end{aligned}
$$

From here we write out the first variational system out as,

$$
\begin{aligned}
\mp\partial_t w^\pm &= \gamma\partial_x w^\pm + \Lambda(w^+ - w^-) + \frac{1}{2}\lambda_2(w^+\tanh(y-y_0) - w^-\tanh(-y-y_0)) + \\
&\quad \frac{1}{2}\lambda_2(u^+\mathrm{sech}^2(y-y_0) + u^-\mathrm{sech}^2(-y-y_0))\left(dq_\ell(K_\ell \star u^- - K_\ell * u^+) + y_w\right), \qquad (4.4.4)\\
\partial_t dq_\ell &= 0,
\end{aligned}
$$

where

$$
y_w = \sum_{i=\{a,r,al\}} q_i(K_i \star w^- - K_i * w^+).
$$

The last thing we note is that since the first variational equation is dependent on the current state and parameter, we have to time step equation (2.3.5) simultaneously with equation (4.4.4), therefore we state the full system to time-step,

$$
\begin{aligned}
\mp\partial_t u^\pm &= \gamma\partial_x u^\pm + \Lambda(u^+ - u^-) + \frac{1}{2}\lambda_2(u^+\tanh(y-y_0) - u^-\tanh(-y-y_0)),\\
\partial_t q_\ell &= 0,\\
\mp\partial_t w^\pm &= \gamma\partial_x w^\pm + \Lambda(w^+ - w^-) + \frac{1}{2}\lambda_2(w^+\tanh(y-y_0) - w^-\tanh(-y-y_0)) + \qquad (4.4.5)\\
&\quad \frac{1}{2}\lambda_2(u^+\mathrm{sech}^2(y-y_0) + u^-\mathrm{sech}^2(-y-y_0))\left(dq_\ell(K_\ell \star u^- - K_\ell * u^+) + y_w\right),\\
\partial_t dq_\ell &= 0.
\end{aligned}
$$

4.4.2 Time-stepping the first variational equation

Now we have to discretize temporally and time-step equation (4.4.5). However, this is simpler now that we have done the work to get equation (4.1.6). If we let

$$\begin{aligned} NLT^{(2)} = &(w^+ \tanh(y-y_0) - w^- \tanh(-y-y_0)) + \\ &(u^+ \mathrm{sech}^2(y-y_0) + u^- \mathrm{sech}^2(-y-y_0)) \left(dq_\ell (K_\ell \star u^- - K_\ell * u^+) + y_w \right), \end{aligned}$$

then we can write equation (4.4.5) as

$$\begin{aligned} \mp \partial_t u^\pm &= \gamma \partial_x u^\pm + \Lambda(u^+ - u^-) + \frac{1}{2}\lambda_2 NLT, \\ \partial_t q_\ell &= 0, \\ \mp \partial_t w^\pm &= \gamma \partial_x w^\pm + \Lambda(w^+ - w^-) + \frac{1}{2}\lambda_2 NLT^{(2)}, \\ \partial_t dq_\ell &= 0. \end{aligned} \tag{4.4.6}$$

and at this point it becomes apparent that since the linear terms have the same form then the application of the Fourier transform and the temporal discretization will result in the same system to solve, only different in the nonlinear terms. We then denote

$$\hat{w}_k^\pm = \widehat{r_{w_k}}^\pm + i\widehat{c_{w_k}}^\pm$$

and

$$\widehat{NLT^{(2)}}_k = r\widehat{NLT^{(2)}}_k + ic\widehat{NLT^{(2)}}_k, \quad k = 0, \ldots, \frac{N}{2}.$$

Then define

$$\Omega^* = \begin{pmatrix} \Omega & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \Omega & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}, \quad Q^* = \begin{pmatrix} Q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & Q & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{aligned} NLTv^* = \Bigg(&\widehat{rNLT}^T, \widehat{cNLT}^T, \widehat{rNLT}^T, \widehat{cNLT}^T, 0, \\ &\left(\widehat{rNLT}^{(2)}\right)^T, \left(\widehat{cNLT}^{(2)}\right)^T, \left(\widehat{rNLT}^{(2)}\right)^T, \left(\widehat{cNLT}^{(2)}\right)^T, 0 \Bigg)^T, \end{aligned}$$

where

$$\widehat{rNLT}^{(2)} = \left(\widehat{rNLT}_0^{(2)}, \ldots, \widehat{rNLT}_{\frac{N}{2}}^{(2)}\right)^T, \quad \widehat{cNLT}^{(2)} = \left(\widehat{cNLT}_1^{(2)}, \ldots, \widehat{cNLT}_{\frac{N}{2}-1}^{(2)}\right)^T.$$

Now let

$$U^* = \left((\hat{r}^+)^T, (\hat{c}^+)^T, (\hat{r}^-)^T, (\hat{c}^-)^T, q_\ell, (\hat{r_w}^+)^T, (\hat{c_w}^+)^T, (\hat{r_w}^+)^T, (\hat{c_w}^+)^T, dq_\ell\right)^T$$

and then we can go from a currently known state $U^*(t_m) = U_m^*$ to the next time-step $U^*(t_{m+1}) = U_{m+1}^*$ by solving the system of equations,

$$(2\Omega^* - \Delta Q^*)U_{m+1}^* = (2\Omega^* + \Delta Q^*)U_m^* + \Delta\lambda_2 NLT v^*. \tag{4.4.7}$$

The last thing we need to be able to time-step equation (4.4.5) is have a way to compute $\widehat{NLT}^{(2)}$. However this also comes almost entirely from the work earlier to compute $\widehat{NLT}$. We restate,

$$\begin{aligned} NLT^{(2)} = &(w^+ \tanh(y - y_0) - w^- \tanh(-y - y_0)) + \\ &(u^+ \text{sech}^2(y - y_0) + u^- \text{sech}^2(-y - y_0)) \left(dq_\ell(K_\ell \star u^- - K_\ell * u^+) + y_w\right), \end{aligned}$$

and then let

$$y_\ell = K_\ell \star u^- - K_\ell * u^+.$$

We know how to compute y and with only slight variations we state the algorithms to compute $\widehat{y_w}$ and $\widehat{y_\ell}$ as,

$$\begin{aligned} \Re(\widehat{y_w}) &= C_k(\hat{r_w}_k^- - \hat{r_w}_k^+) - S_k(\hat{c_w}_k^+ + \hat{c_w}_k^-), \\ \Im(\widehat{y_w}) &= C_k(\hat{c_w}_k^- - \hat{c_w}_k^+) + S_k(\hat{r_w}_k^- + \hat{r_w}_k^+), \\ \Re(\widehat{y_\ell}) &= \frac{1}{q_\ell} C_{k,\ell}(\hat{r_w}_k^- - \hat{r_w}_k^+) - \frac{1}{q_\ell} S_{k,\ell}(\hat{c_w}_k^+ + \hat{c_w}_k^-), \\ \Im(\widehat{y_\ell}) &= \frac{1}{q_\ell} C_{k,\ell}(\hat{c}_k^- - \hat{c}_k^+) + \frac{1}{q_\ell} S_{k,\ell}(\hat{r}_k^- + \hat{r}_k^+). \end{aligned}$$

Once we have $\widehat{y_w}$ and $\widehat{y_\ell}$ we take them to real space and get y_w and y_ℓ. We would then take $\hat{u}^\pm$ and $\hat{w}^\pm$ to real space and get $u^\pm$ and $w^\pm$. Again we would pad the Fourier space representations of these variables with zeroes before taking them to real space if we were attempting to de-alias the nonlinearity. Then we can compute $NLT^{(2)}$ in real space and take it back to Fourier space with our DFT and get $\widehat{NLT}^{(2)}$.

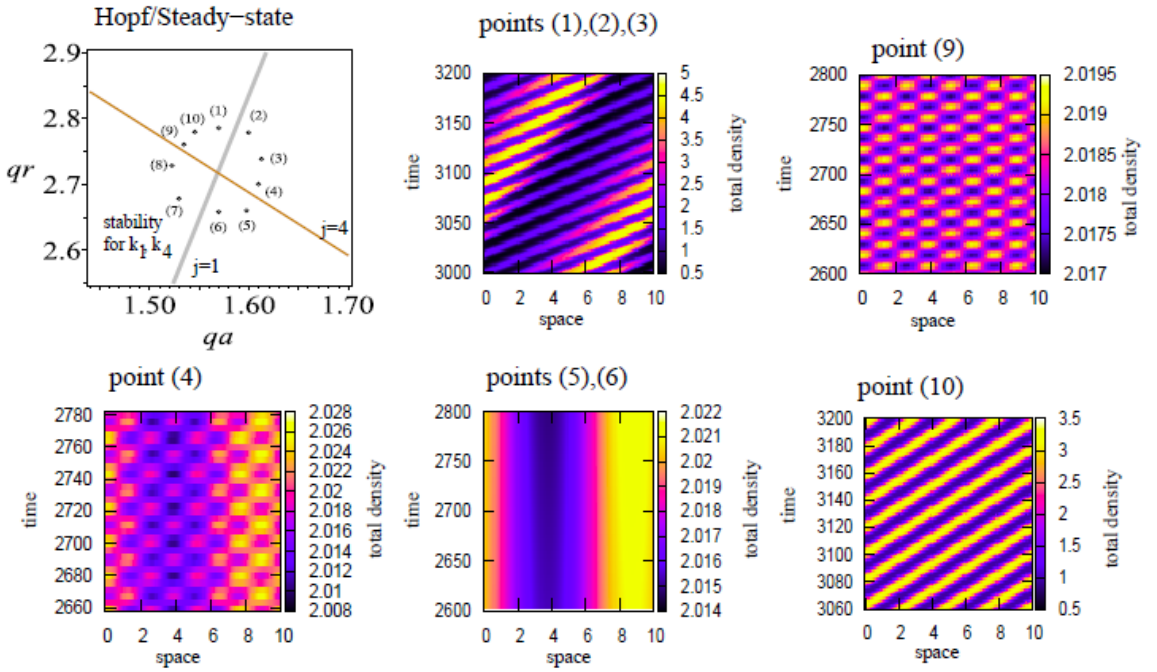

Figure 4.2: Several points around a Hopf-steady state bifurcation curve crossing (top left) along with dynamics observed. Plots show the total density at a point in space and time. Taken from Buono and Eftimie [4].

4.5 Validation tests of time-steppers

The simplest tests we can do to check if our time-steppers are performing correctly is to choose parameter values , q_i for $i \in \{a, r, al\}$, which we know what the solutions should look like. Figure 4.2 shows several points in (q_a, q_r)-space with $q_{al} = 0$ around the crossing of Hopf and steady state bifurcation curves along with solutions observed from simulations from [4] at some of the points.

We run simulations to the same final times as those in Figure 4.2 with approximated parameter values based on the figure. Figure 4.3 shows the results of our simulations.

The slight difference observed in the simulation of point 4 is not significant as this particular structure of the solution is unstable and the scale of the wave structure under the bumps is $\mathbf{O}(10^{-3})$. The difference in the simulation of point 10 could be due to an error in the approximation of the parameter values from Figure 4.2, though

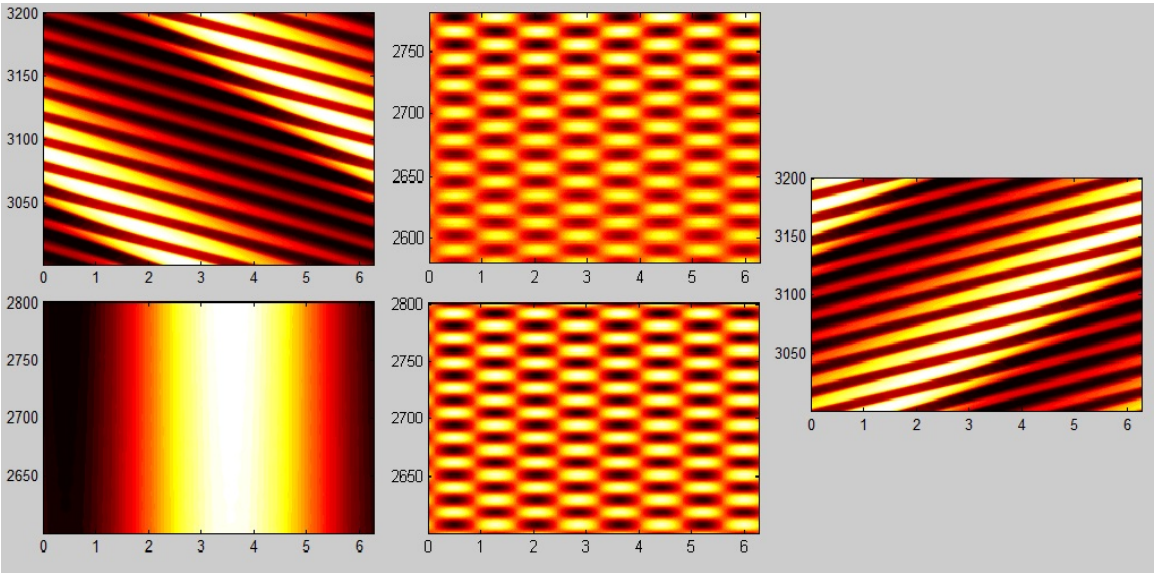

Figure 4.3: Simulations of point 1 (top left), point 4 (top middle), point 6 (bottom left), point 9 (bottom middle), and point 10 (right). Space is on the x-axis with time on the y-axis. Plots show total density at a point in space and time. Note the resemblance to dynamics observed in Figure 4.2.

this is the most striking difference from the points. However, these differences are not proof of the code not functioning properly because initial conditions are chosen randomly and for some parameter values there may exist more than one meta-stable solutions, which becomes more complicating with the long evolution times required to see convergence to states. Besides these particular points, Figure 4.4 shows figures from our simulations which show some of the same dynamics seen in Figure 2.4. These are not necessarily for the same parameters, integration times, or other numerical parameters, but seeing in our simulations the same dynamics seen in simulations by Eftimie *et al.* [5] is reassuring.

Our evolutions of equation (2.3.5) showcase similar dynamics for the same parameter values as shown by Buono and Eftimie [4]. Additionally it showcases a few of the more exotic structures like zigzag and feather patterns as well as the more basic structures like triple pulses.

Turning our attention to evolving equation (4.4.5), we remember from Figure 2.5 that we know values in (q_a, q_r)-space such that the homogeneous steady state is stable.

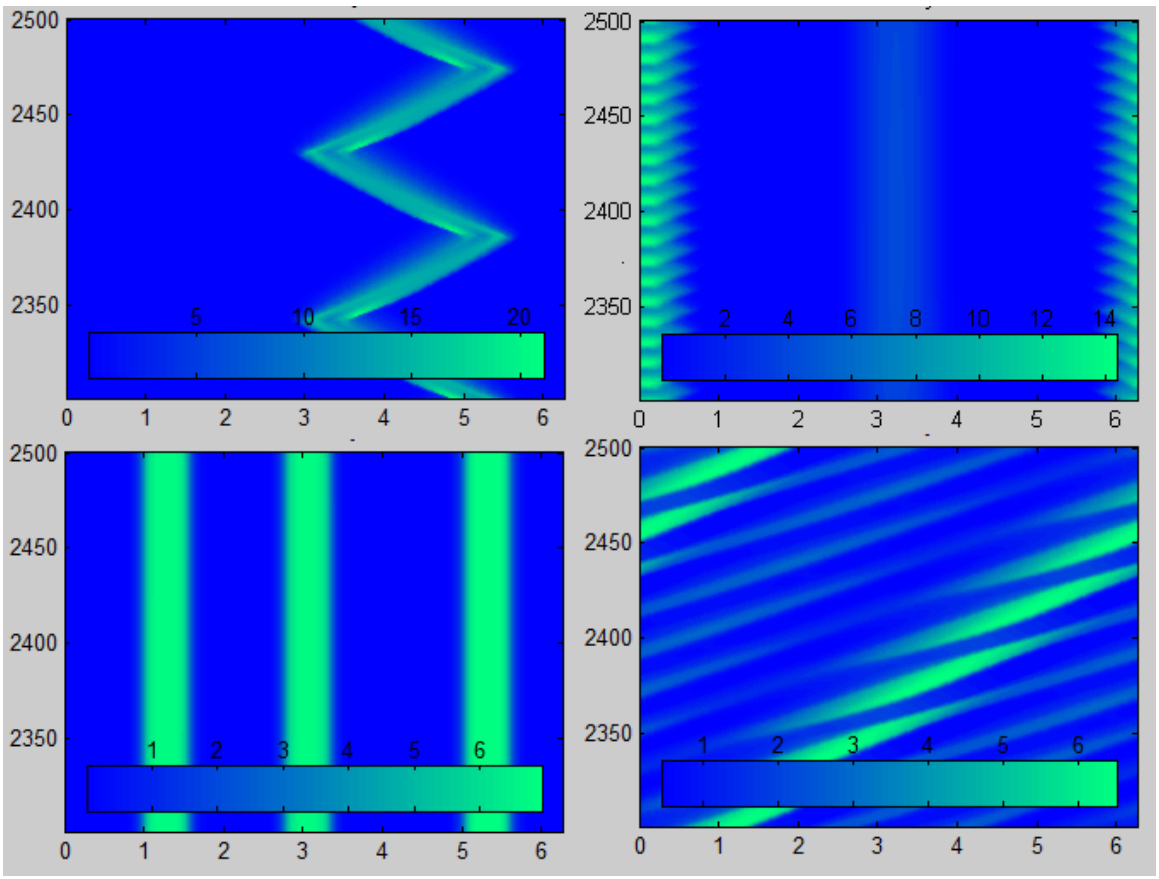

Figure 4.4: Several dynamics from our simulations showcasing similarity to dynamics observed in Figure 2.4. We can see a zigzag pulse (top left), a pattern similar to feathers (top right), three stationary pulses (bottom left), and breathers (bottom right).

Therefore we can test our evolution of equation (4.4.5) using this information. For these tests we choose initial conditions,

$$\begin{aligned} u_0^+ &= 1_{N,1}, \\ u_0^- &= 1_{N,1}, \\ w_0^+ &= v, \quad v \in X_N, \\ w_0^- &= z, \quad z \in X_N. \end{aligned}$$

In addition, we will pick $\ell = al$ but this is a trivial choice anyway as we will be choosing initial parameter conditions,

$$q_{\ell,0} = 0, \quad dq_{\ell,0} = 0.$$

The first test will run with $(q_a, q_r) = (-1, 2)$ while the second test will run with $(q_a, q_r) = (-3, 3)$. The first test is in the region of stability for the homogeneous steady state so we should see the perturbations decay. The second test is in the region of instability so we should see perturbations increase. Figure 4.5 shows the results from this test and you see the expected behaviors do occur in our simulations.

So our tests show that we are getting the right behavior from our simulations but we have not shown that the methods themselves are working correctly. Therefore we want to do more tests. The first test we will do is on the time-stepper for equation (4.1.6). We will test the dependence of an approximation of the error between the true solution and the numerical solution we have on the number of grid points and the size of time-steps.

The basic scheme we will use is; choose an initial condition u_0. Let us call $u_{N,\Delta}$ the solution from evolving u_0 to a total time of 20 with N grid points and Δ time-step size. Then we compute the approximate errors,

$$Err_N^{(1)} = ||u_{N,\Delta} - u_{\frac{N}{2},\Delta}||_{L_2}, \quad Err_\Delta^{(2)} = ||u_{N,\Delta} - u_{N,10\Delta}||_{L_2},$$

so that $Err_N^{(1)}$ is the difference between two solutions, with one having twice the grid points of the other. So $Err_N^{(1)}$ will tell us how the error approximately depends on the number of grid points. For pseudo-spectral methods we expect the error to decrease at

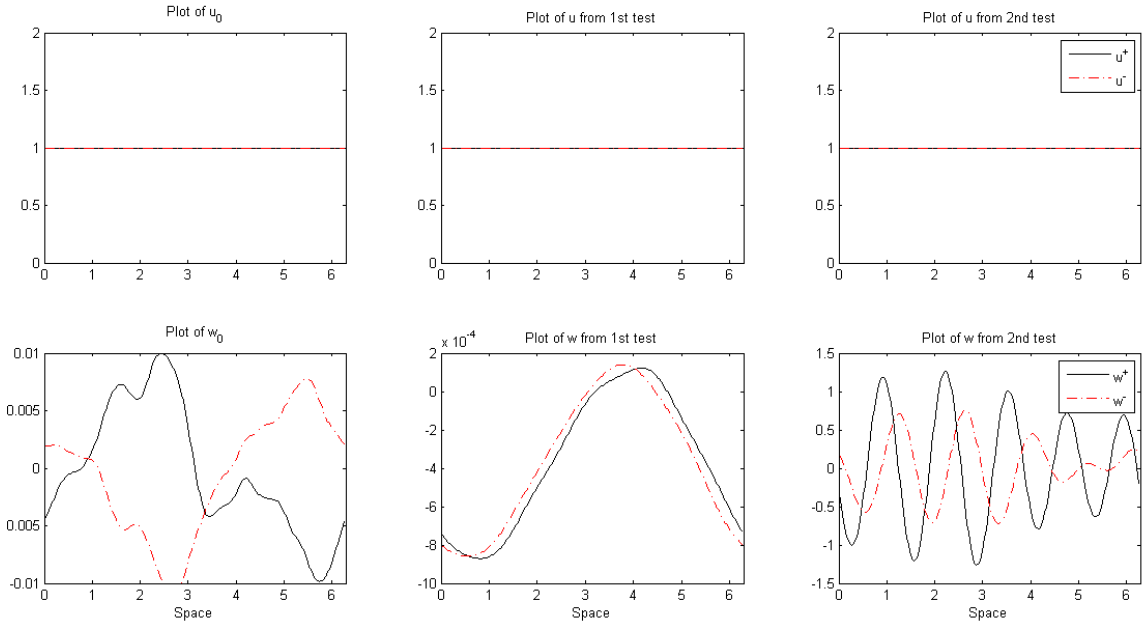

Figure 4.5: Initial conditions of tests of the time-stepper for equation (4.4.5) (left), results from test within the stability region of the homogeneous steady state (middle), and results from test outside the stability region of the homogeneous steady state (right). Bottom plots show the density distributions of perturbations and top plots show the density distributions of populations.

least exponentially fast as the number of grid points increases. $Err^{(2)}_{\Delta}$ is the difference between two solutions, with one having an order of magnitude larger of a time-step than the other. So $Err^{(2)}_{\Delta}$ will tell us how the error approximately depends on the time-step size. Since we used a forward Euler approximation in the temporal discretization we expect $\mathbf{O}(\Delta)$ dependence.

In order to compute $Err^{(1)}$ we need to compute the difference between vectors which are defined on different numbers of grid points. This problem is dealt with in noticing that since the number of grid points take the form $N = 2^m$ then if we consider a solution on 2^p grid points and another on 2^q grid points, assuming $q < p$ without loss of generality, then their spatial grids will align exactly every 2^{p-q} grid points so we merely compute the difference between the solutions on these shared grid points.

This process was performed with three different initial conditions; the first are homogeneous states,

$$u_0^+ = 1_{N,1} + v, \quad u_0^- = 1_{N,1} + z, \quad v, z \in X_N,$$

the second are inhomogeneous constant-valued states,

$$u_0^+ = \frac{1}{2}1_{N,1} + v, \quad , u_0^- = \frac{3}{2}1_{N,1} + z, \quad v, z \in X_N,$$

and the third are squared sine and cosine waves,

$$u_0^+ = \sin^2(x), \quad u_0^- = \cos^2(x), \quad x = \left(0, \frac{2\pi}{N}, \ldots, \frac{2\pi}{N}(N-1)\right)^T.$$

In addition, each of these initial conditions is tested with two different sets of parameters,

$$(q_a, q_r, q_{al}) = (-1, 2, 0), \quad (q_a, q_r, q_{al}) = (-1.3, 2.1, 3.6).$$

Figures 4.6, 4.7, 4.8, 4.9, 4.10, 4.11 shows the results of these tests.

You can see in all cases we see a super exponential decrease in the approximated error as the number of grid points increases linearly and a linear decrease in error as the time-step size decreases linearly, although for some cases this is only true below a certain time-step size but this is reasonable. This super exponential decrease, above what is minimally expected, could either be because the filter chosen on the initial condition makes it almost C^∞ smooth and that is what we're seeing, or the actual

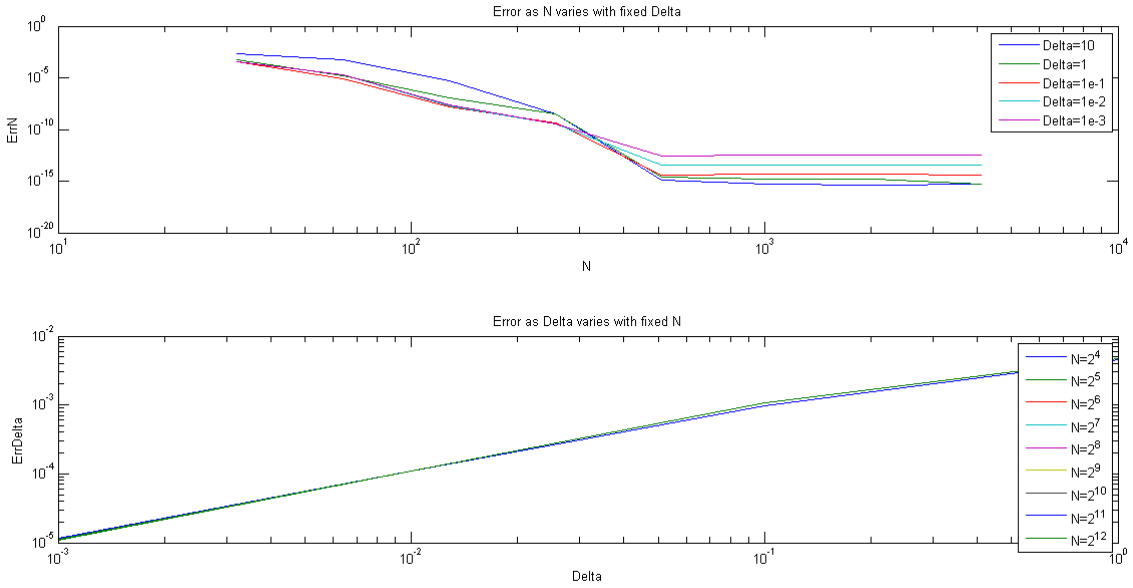

Figure 4.6: Error dependence on the number of grid points (top) along with error dependence on the time-step size (bottom) for homogeneous states with 0.01 amplitude perturbations and $(q_a, q_r, q_{al}) = (-1, 2, 0)$.

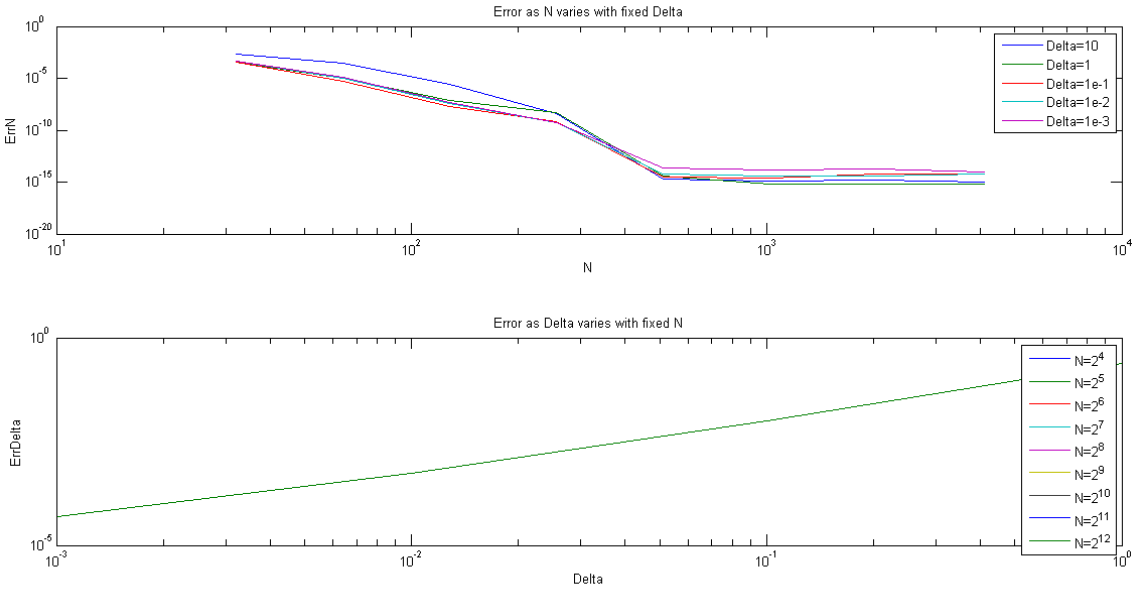

Figure 4.7: Error dependence on the number of grid points (top) along with error dependence on the time-step size (bottom) for inhomogeneous constant-valued states with 0.01 amplitude perturbations and $(q_a, q_r, q_{al}) = (-1, 2, 0)$.

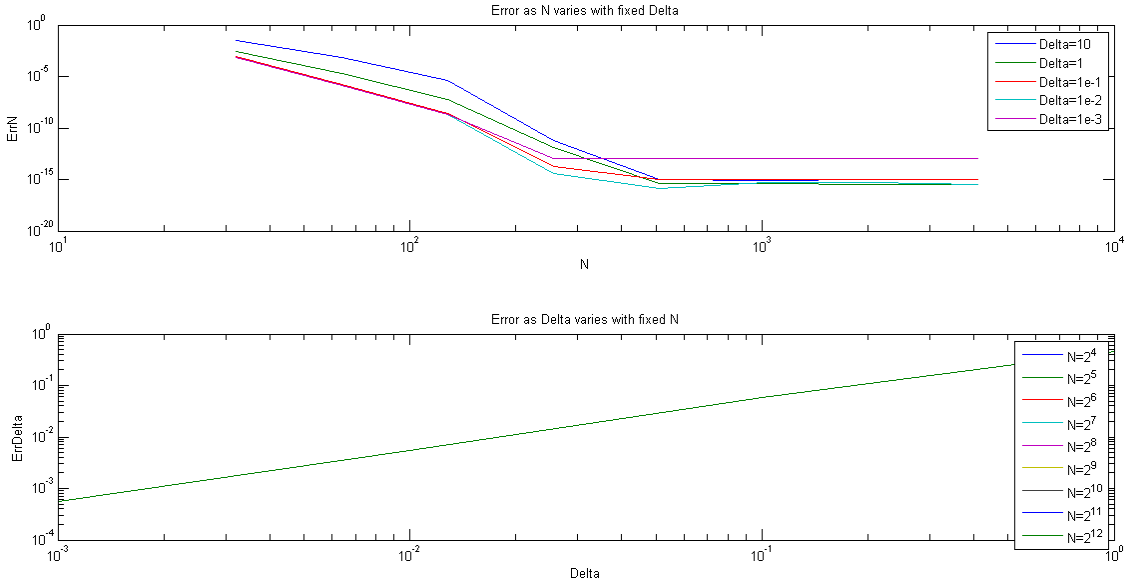

Figure 4.8: Error dependence on the number of grid points (top) along with error dependence on the time-step size (bottom) for squared sine and cosine states and $(q_a, q_r, q_{al}) = (-1, 2, 0)$.

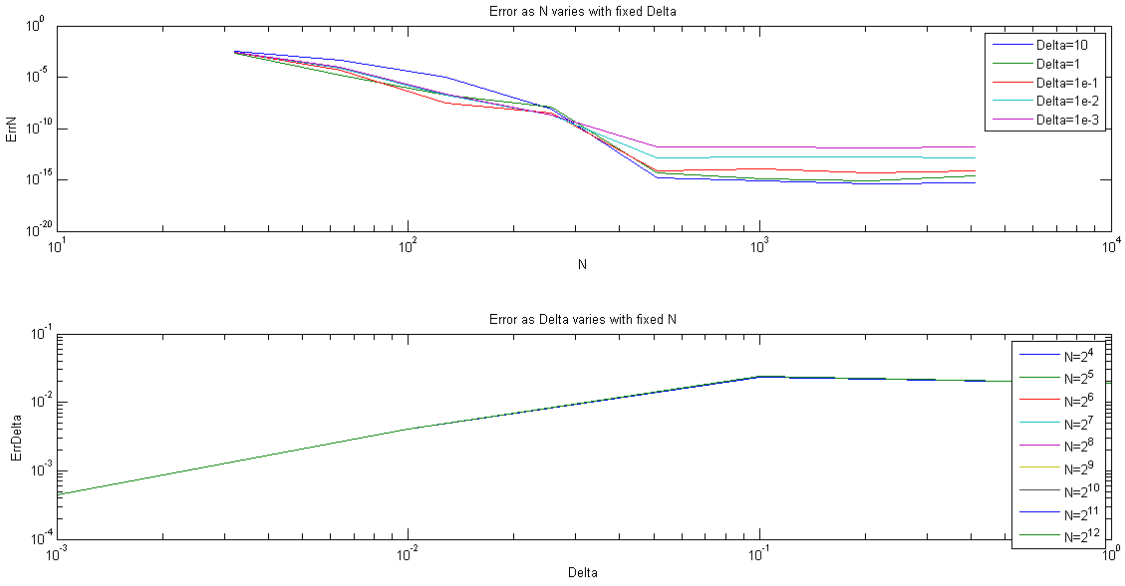

Figure 4.9: Error dependence on the number of grid points (top) along with error dependence on the time-step size (bottom) for homogeneous states with 0.01 amplitude perturbations and $(q_a, q_r, q_{al}) = (-1.3, 2.1, 3.6)$.

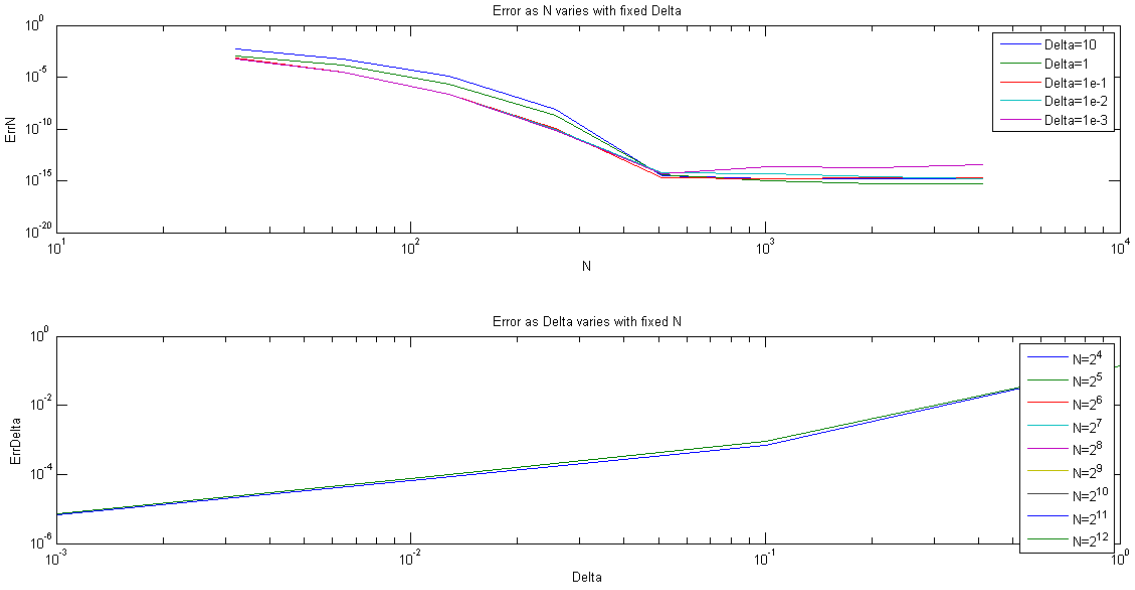

Figure 4.10: Error dependence on the number of grid points (top) along with error dependence on the time-step size (bottom) and for inhomogeneous, constant-valued states with 0.01 amplitude perturbations and $(q_a, q_r, q_{al}) = (-1.3, 2.1, 3.6)$.

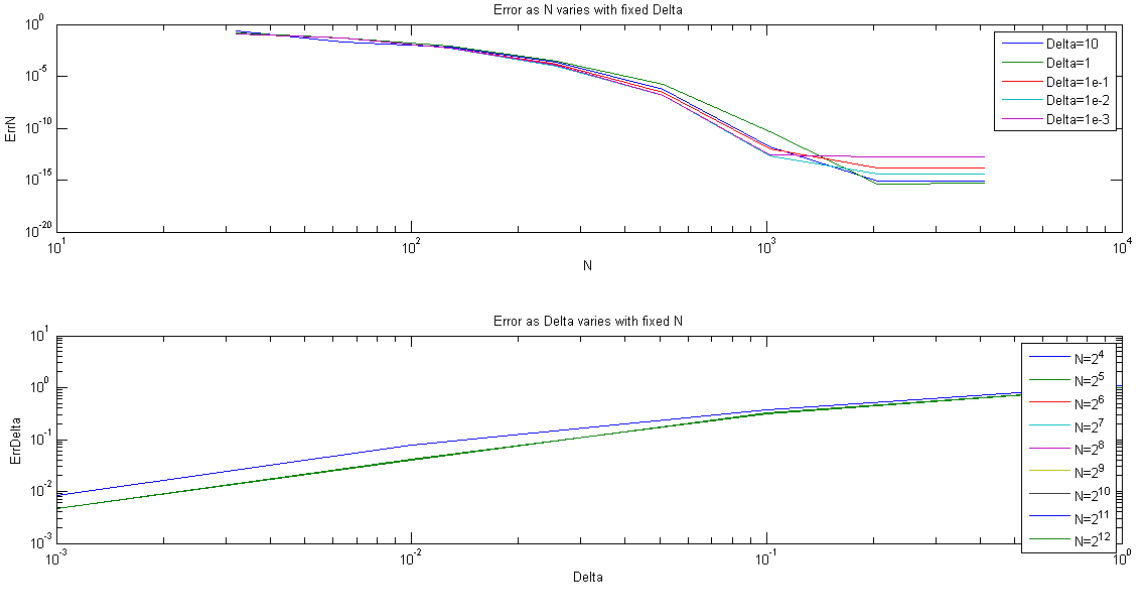

Figure 4.11: Error dependence on the number of grid points (top) along with error dependence on the time-step size (bottom) and for squared sine and cosine states and $(q_a, q_r, q_{al}) = (-1.3, 2.1, 3.6)$.

solution we are converging to is in fact C^∞. For the squared sine and cosine initial conditions the smoothness of the initial condition is assured. The time-stepper is functioning correctly.

Next we look to the Taylor series expansion we used earlier to derive the first variational equation,

$$f(u+\epsilon w, q_\ell + \epsilon dq_\ell) \approx f(u, q_\ell) + \epsilon Df(u, q_\ell)(w, dq_\ell)^T.$$

Rearranging we are specifically interested in,

$$\frac{f(u+\epsilon w, q_\ell + \epsilon dq_\ell) - f(u, q_\ell)}{\epsilon} \approx Df(u, q_\ell)(w, dq_\ell)^T.$$

The left-hand side can be computed by evolution of equation (2.3.5) and the right-hand side can be computed by evolution of equation (4.4.5). Therefore we will get the best approximation with respect to ϵ of the left-hand side and compare it to what we get for the right-hand side. We begin with the first task at hand. Also let us choose $\ell = al$.

We choose initial conditions $u_0 = (1_{1,N}, 1_{1,N})^T$ and $w_0 = (v, z), \quad v, z \in X_N$ then set

$$w_0 \to \frac{w_0}{||w_0||_{L_2}}.$$

Also let $dq_\ell = 1$. Then let us call u the solution after evolving the initial condition u_0 to a total time of 160 according to equation (2.3.5) with parameters $(q_a, q_r, q_{al}) = (-1.3, 2.1, 3.6)$. Call $u^{(m)}$ the solution after evolving the initial condition $u_0 + 10^{-m}w_0$ to a total time of 160 according to equation (2.3.5) with parameters $(q_a, q_r, q_{al}) = (-1.3, 2.1, 3.6 + 10^{-m})$. Then

$$\frac{f(u+\epsilon w, q_\ell + \epsilon dq_\ell) - f(u, q_\ell)}{\epsilon} \approx \frac{u^{(m)} - u}{10^{-m}}$$

and we define the approximate error,

$$Err_m^{(3)} = \left|\left| \frac{u^m - u}{10^{-m}} - \frac{u^{m-1} - u}{10^{-(m-1)}} \right|\right|_{L_2}, \quad m = 1, \ldots, 16.$$

The best approximation to $Df(u, q_\ell)(w, dq_\ell)^T$ will be when this error takes its minimum. Let m^* be such that $Err_{m^*}^{(3)}$ is this minimum. Then call u^* and w^* the solution to equation (4.4.5), neglecting the parameter and perturbation to the parameter, with

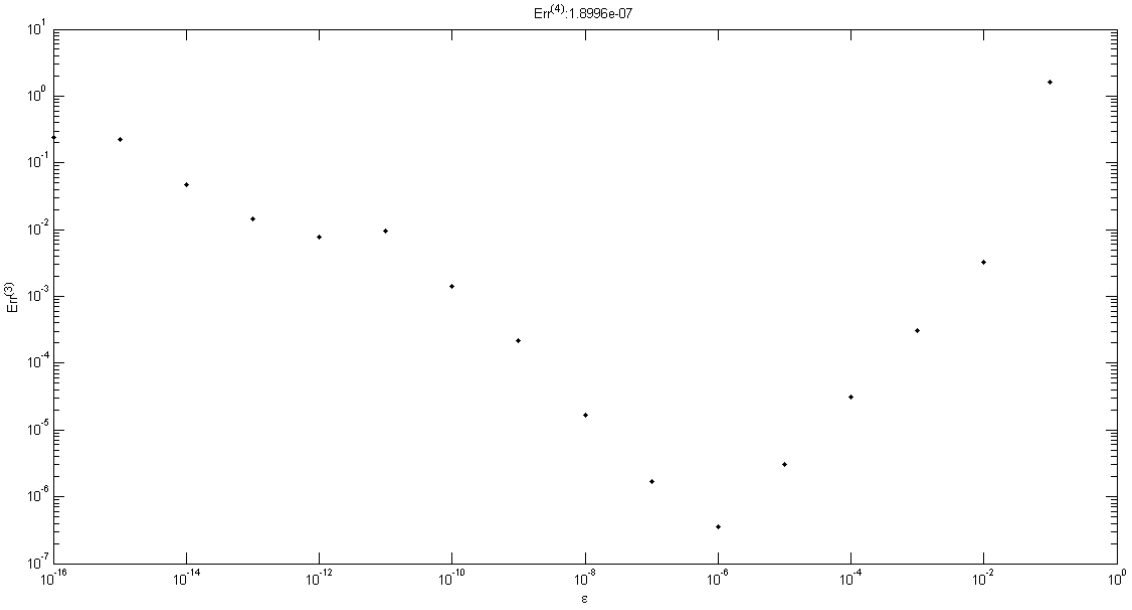

Figure 4.12: Approximate errors of finite difference approximations to $Df(w, dq_\ell)^T$ along with comparison of best finite difference approximation to result from time-stepping equation (4.4.5) showing comparable errors.

initial conditions u_0, $q_\ell = 3.6$, w_0, and $dq_\ell = 1$. Then form,

$$Err^{(4)} = \left|\left| \frac{u^{m^*} - u}{10^{-m^*}} - w^* \right|\right|_{L_2}.$$

It is expected for $Err^{(3)}$ to initially decrease linearly as $\epsilon = 10^{-m}$ decreases until the approximation stops refining, numerical noise dominates, and the error climbs back up. Indeed this is what we see in Figure 4.12. $Err^{(3)}$ decreases linearly with respect to ϵ to a point and $Err^{(4)}$ is on the same order as the best finite difference approximation. The time-steppers are working properly judging by our tests.

Chapter 5
Continuation methods

5.1 Motivation and framework

Continuation methods begin with a known solution to a system of equations and traces out a curve of solutions from your known solution, where each point on the solution curve corresponds to a different solution with potentially different parameter values. Figure 5.1 gives an example of one such curve with depictions of the dynamics at various points. Notice there is a branching point where another curve of solutions branches at a critical parameter value. The y-axis is typically chosen to measure some property of the solution but is open to choice. In our case we let the y-axis depict the amplitude of the solution. The purpose of this depiction is to get an idea of what types of solutions can be seen for certain parameter values.

In practical applications the system of equations mentioned, which we will denote as a general condition $g(u, \lambda)$, is typically a set of algebraic equations or boundary value problems that describe some specific phenomenon. For our purpose these phenomena can be equilibria, periodic orbits, or other more complicated invariant objects of equation (2.3.5). The curves as in Figure 5.1 then characterize the types of dynamics that can be observed from long-term simulations and are powerful tools from a practical analysis perspective. In the continuation methods we seek to make guesses from approximately known points satisfying g and then refine these guesses until they satisfy g well enough.

As mentioned in Chapter 3, time-stepping equation (2.3.5) until we converge to a state can take long evolution times. What continuation methods give us then is the ability to start from a state that is assumed converged and follow the evolution of this state as a parameter is varied. If we wanted to see the dynamics exhibited by a

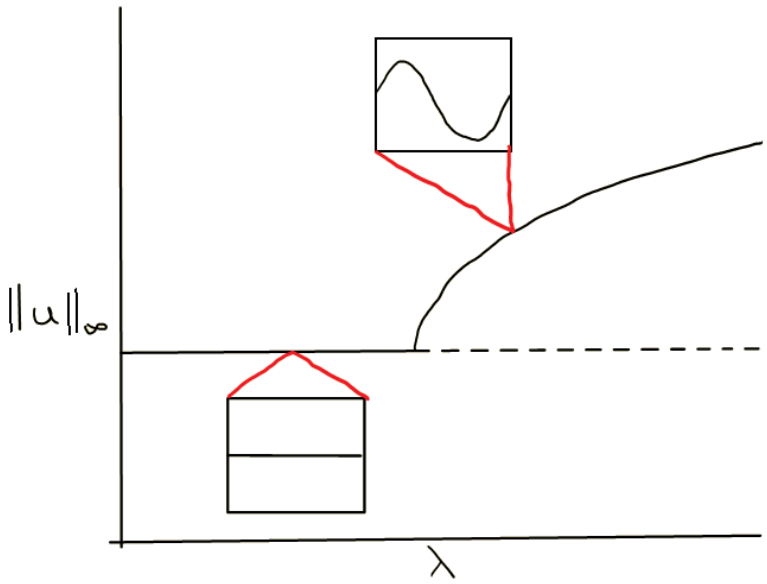

Figure 5.1: A curve of solutions along with another curve branching from it at some critical parameter value. Solid lines represent a stable curve while dashed lines represent an unstable curve. We observe a stable homogeneous steady state bifurcate at some critical parameter value where another state becomes stable with an amplitude that increases as the parameter is increased

state for a particular set of parameters with our time-steppers we would have to set parameter values, initialize our typical initial condition, and evolve for a potentially long time. Furthermore, whether or not what we get at the end of this process is converged may also be questionable.

With continuation methods we need only do this process once and we can do it for values of parameters where we know what the state should be, such as the homogeneous steady state in its stability region of parameter space. Our discussion now focuses on equilibrium solutions. After we are pleased with the resolution of our equilibrium we apply continuation methods to see other equilibria for lots of different parameter values and furthermore the process we employ to trace out this curve of equilibria will require evolution times up to three orders of magnitude less than evolution times for getting to the same equilibrium with just the time-stepper.

In general there are two components to any continuation method, a predictor and a corrector. The predictor component gives a guess at the next point in the curve of solutions, typically with extrapolation methods. The corrector component then takes the guess and refines it until some tolerance on the approximate error of the numerical solution and the true solution at that point is satisfied.

Let us denote by (u_k, λ_k) for $k = 0, 1, \ldots$ the points on the curve of solutions which we compute with continuation methods such that (u_0, λ_0) is the initial point given by some other method, which in our case comes from evolution of the equation (2.3.5). Successive points are obtained by one iteration of using the predictor component from the previous point and then refining it with the corrector component. Finally call $(\bar{u}_0, \bar{\lambda}_0)$ the point guessed with the predictor and call $(\bar{u}_k, \bar{\lambda}_k)$ for $k = 1, 2, \ldots, M^*$ successive iterates of the corrector such that M^* is the step at which we satisfy the tolerance on our approximate error.

We will use pseudo-arclength continuation with a Newton corrector as introduced by Keller [8]. The predictor will then make a guess by extending some step-size, Δs, along an approximated tangent, T, of the last known solution point. If u_k, λ_k is our

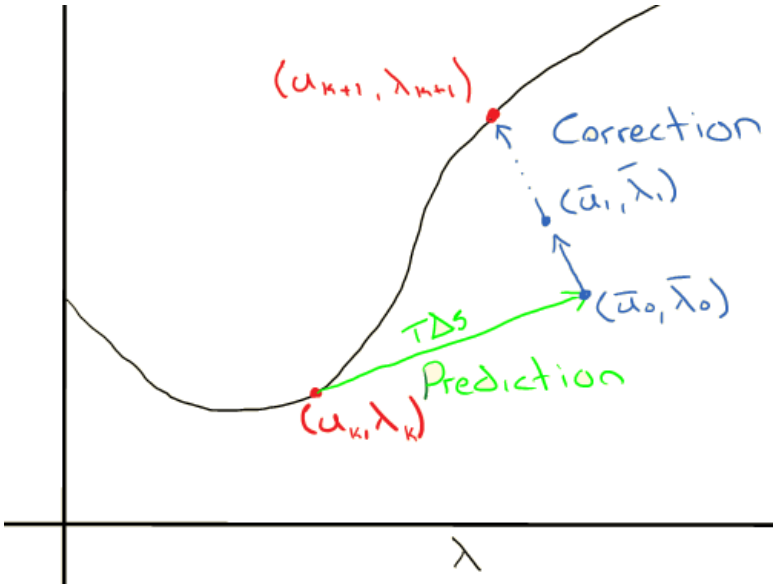

Figure 5.2: Visualization of one iteration of pseudo-arclength continuation with the black curve as the true curve of solutions. Prediction extends a distance Δs along the tangent and then correction iteratively updates the guess in an orthogonal direction until it is close enough in some measure.

last known solution point then

$$(\bar{u}_0, \bar{\lambda}_0) = (u_k, \lambda_k) + T\Delta s.$$

The choice of Δs is problem specific and therefore varies, but a general strategy is to attempt one iteration of the predictor-corrector components at some stepsize, Δs^*, and if the corrector fails to converge then restart that step with a reduced step-size such as $\frac{\Delta s^*}{2}$. Repeat this process until convergence of the corrector is achieved.

The approximated tangent is typically taken as the normalized secant between the last two computed points. However, when continuing from (u_0, λ_0) we generally take the approximated tangent as a unit vector in the direction of the parameter. Figure 5.2 gives a visualization of this process for one iteration.

In general the corrector for pseudo-arclength methods must satisfy some condition, g, such that u is a solution with parameter λ, and an orthogonality condition so that

$(\bar{u}_k, \bar{\lambda}_k)$ are in a direction orthogonal to the tangent. Together these are stated as,

$$g(u, \lambda) = 0, \quad T \cdot (u - \bar{u}_0, \lambda - \bar{\lambda}_0) = 0.$$

We apply Newton's method on this system then to refine our initial guess and get,

$$\begin{pmatrix} D_u g & D_\lambda g \\ T_u^T & T_\lambda \end{pmatrix} \begin{pmatrix} du \\ d\lambda \end{pmatrix} = \begin{pmatrix} -g \\ -T \cdot (u - \bar{u}_0, \lambda - \bar{\lambda}_0) \end{pmatrix}.$$

Formally then, the corrector receives the guess $(\bar{u}_0, \bar{\lambda}_0)$ from the predictor and iteratively generates

$$(\bar{u}_{k+1}, \bar{\lambda}_{k+1}) = (\bar{u}_k, \bar{\lambda}_k) + (du_k, d\lambda_k), \quad k = 0, 1, \ldots, \text{until converged}$$

such that the update satisfies the Newton system

$$\begin{pmatrix} D_u g(\bar{u}_k, \bar{\lambda}_k) & D_\lambda g(\bar{u}_k, \bar{\lambda}_k) \\ T_u^T & T_\lambda \end{pmatrix} \begin{pmatrix} du_k \\ d\lambda_k \end{pmatrix} = \begin{pmatrix} -g(\bar{u}_k, \bar{\lambda}_k) \\ -T \cdot (\bar{u}_k - \bar{u}_0, \bar{\lambda}_k - \bar{\lambda}_0) \end{pmatrix}$$

until the update $(du_k, d\lambda_k)$ or the residual $g(\bar{u}_k, \bar{\lambda}_k)$ are small enough. Again recall $(\bar{u}_{M^*}, \bar{\lambda}_{M^*})$ is then the iterate at which this convergence criteria is achieved, then if (u_q, λ_q) is the last known point on the solution curve then

$$(u_{q+1}, \lambda_{q+1}) = (\bar{u}_{M^*}, \bar{\lambda}_{M^*})$$

and this is the completion of one iteration of the predictor-corrector process and we may iterate again for the next point.

For our purposes we have that $u = (\hat{r}^+, \hat{c}^+, \hat{r}^-, \hat{c}^-)$, $\lambda = q_\ell$, $du = (\hat{r_w}^+, \hat{c_w}^+, \hat{r_w}^-, \hat{c_w}^-)$, and $d\lambda = dq_\ell$. For the condition, g, we vary this depending on the type of state we are looking for. Equilibria require $\dot{u} = 0$ so $q \equiv f$. The condition would change for more complicated states but we brush that aside as we will introduce a different formulation of our condition with the flow operator.

5.2 Use of the flow operator and concern of symmetries

The flow operator evolves a given state to a certain time τ. Formally we say the flow operator, ϕ_τ, acts on states as

$$\phi_\tau u(x, t) = u(x, t + \tau).$$

By using the flow operator in our condition, g, the Jacobian of the flow acting on the perturbations du and $d\lambda$ can be computed by time-stepping the first variational equation. Though it should be mentioned that there are other ways to achieve matrix-free continuation methods, by using the flow operator we need only one evolution to get the matrix-vector product as we will describe in Chapter 5, Section 5.3.

One benefit of using the flow operator is that its Jacobian is better conditioned for the linear solver we will be using; the Generalized Minimal Residual method (GMRES). This is because GMRES functions best when the linear system has a spectrum which is clustered. The reader may refer to Saad and Schultz for an introduction and description of GMRES [12]. How does this relate to the Jacobian of the flow operator? Well, if λ is an eigenvalue of Df and u is an equilibrium or periodic state then $\exp(\tau\lambda)$ is an eigenvalue of the Jacobian of ϕ_τ. So if equation (2.3.5) is dissipative then most of its eigenvalues will have negative real part and then the spectrum of our flow operator will cluster near the origin.

However, equation (2.3.5) has not been shown to be dissipative. Figure 5.3 shows us the spectrum of the instantaneous Jacobian of equation (2.3.5) for four converged states. You can see for simple states like the homogeneous steady state the entire spectrum has negative real part but for more complicated states the spectrum starts to get more and more eigenvalues with positive real part. It would not be unreasonable to think that for even more complicated states the spectrum of the instantaneous Jacobian could have even more eigenvalues with positive real part and the more eigenvalues with positive real part the less clustered the spectrum of the Jacobian of the flow will be, making the system less well conditioned.

With the flow operator we can restate our condition for a converged state of the system. Thus our condition will be

$$g \equiv \phi_\tau u - u.$$

So our Newton system becomes,

$$\begin{pmatrix} D_u\phi_\tau - I & D_\lambda\phi_\tau \\ T_u^T & T_\lambda \end{pmatrix} \begin{pmatrix} du \\ d\lambda \end{pmatrix} = \begin{pmatrix} -\phi_\tau u + u \\ -T \cdot \left((u - \bar{u}_0, \lambda - \bar{\lambda}_0)\right) \end{pmatrix}.$$

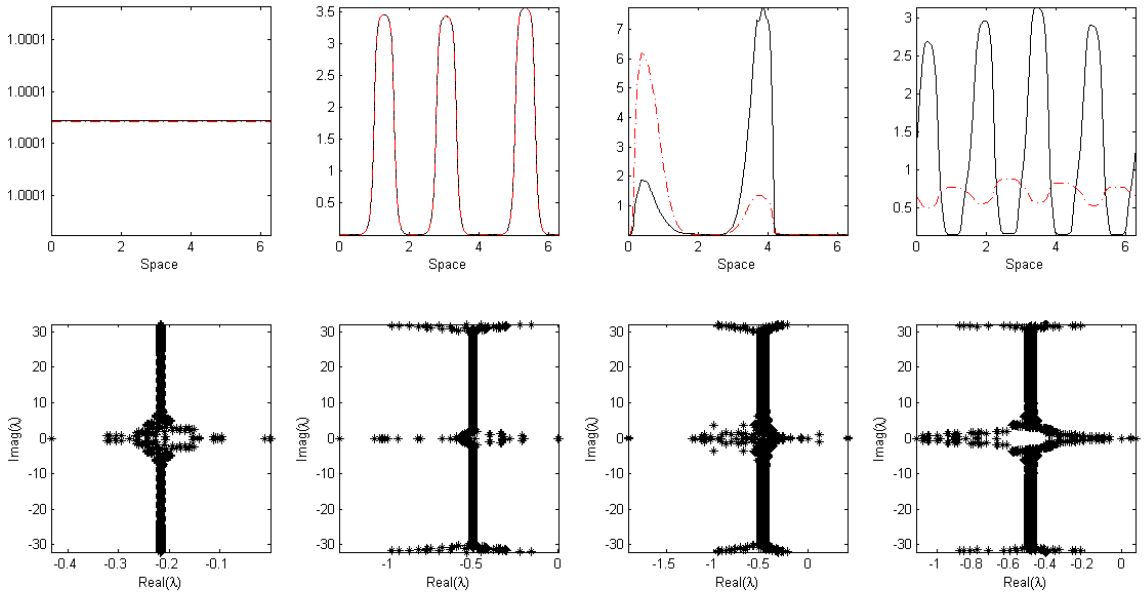

Figure 5.3: Four states shown top as their final time density distributions of populations and the spectrum for perturbations of the instantaneous Jacobian of equation (2.3.5) about these states on the bottom. Notice that for more exotic dynamics the spectrum gains more eigenvalues with positive real part.

At this point we need to address an issue that can arise with the corrector. Since the update step in our corrector arises from the solution of the Newton system, degeneracies in $D\phi_\tau$ can cause problems with the update. If there exists a neutral direction such that an eigenvalue of Df is zero, or equivalently an eigenvalue of $D\phi_\tau$ is one, other than the flow direction then the update may choose to move the solution in that direction.

The most obvious place this may arise is with the translational symmetry we mentioned in Section 2. If the corrector is trying to update our approximate solution point then in the worst case scenario, the update may be exactly in the direction of the group orbit generated by the translational symmetry. The corrector will continue translating the solution, never converging, and then our method will fail.

We can remove the degeneracy of the translation symmetry by adjusting our condition as

$$g \equiv \phi_\tau u - \Theta_y u$$

and including the translation y into part of the Newton system. We add one final condition as a row on the Newton system to ensure the update is orthogonal to the generator of translations, u_x. This makes our Newton system,

$$\begin{pmatrix} D_u\phi_\tau - \Theta_y & D_\lambda\phi_\tau & \Theta_y u_x \\ T_u^T & T_\lambda & T_y \\ u_x & 0 & \alpha \end{pmatrix} \begin{pmatrix} du \\ d\lambda \\ dy \end{pmatrix} = \begin{pmatrix} -\phi_\tau u + \Theta_y u \\ -T \cdot \left(u - \bar{u}_0, \lambda - \bar{\lambda}_0, y - \bar{y}_0\right) \\ 0 \end{pmatrix}.$$

The α in the matrix is there to prevent degeneracy when $u_x \equiv 0$ and to better condition the 3rd row if necessary. There are other degeneracies that exist in $D\phi_\tau$ which we will go over later in this section but there are not as well understood as the translational degeneracy. Now we have the framework of the continuation method set up.

5.3 Matrix-free continuation methods

At this point all we need to do to complete the continuation method is define an algorithm to accurately solve the Newton system for the updates. We could use traditional solving algorithms that solve the general linear system $Ax = b$, however some of these

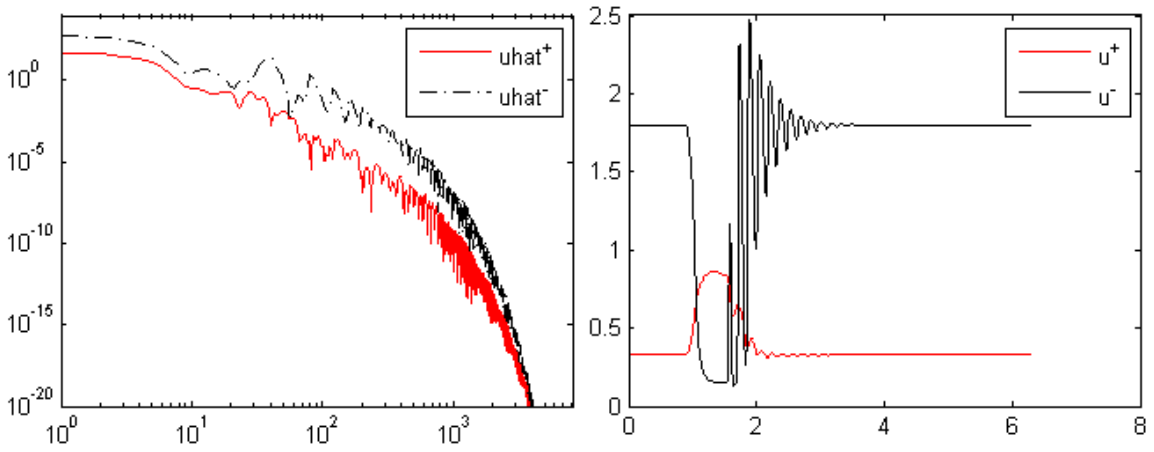

Figure 5.4: Power spectrum of solution (left) along with final time density distribution plot of populations (right). Notice the large number of grid points required to resolve the power spectrum well.

use the full matrix and as such require building, storing, and manipulating the entire matrix. Figure 5.4 shows us that for some choices of the parameter values, we require spatial resolution upwards of $N = 2^{14}$ which would result in our Newton system being 32770 by 32770. For each new state point we wish to find with continuation methods, we can expect about 10 Newton iterations from current simulations, each requiring us to rebuild and manipulate these matrices. Simulations tracing out curves have not been done yet so we have no idea how many new state points will need to be computed in order to draw significant curves of states in parameter space.

As we mentioned, we will be using GMRES. In addition to the conditioning remarks we went over before, GMRES can solve the system without the full matrix A. To solve the linear system all GMRES needs is the matrix-vector product Ax and the constant vector b.

If we have a way to compute the matrix-vector product, Ax, without actually forming the matrix A then we have a way to store only vectors for our Newton step. This is an incredible storage reduction that could speed up the Newton iterations when compared to traditional methods which use methods that build the matrix.

So now we explicitly form the matrix-vector product,

$$\begin{pmatrix} D_u\phi_\tau - \Theta_y & D_\lambda\phi_\tau & \Theta_y u_x \\ T_u^T & T_\lambda & T_y \\ u_x & 0 & \alpha \end{pmatrix} \begin{pmatrix} du \\ d\lambda \\ dy \end{pmatrix},$$

term by term until we have an explicit algorithm for the matrix-vector product in full. We state the matrix-vector product as

$$\begin{aligned} (D_u\phi_\tau du + D_\lambda\phi_\tau d\lambda) - \Theta_y du + \Theta_y u_x dy, \\ T \cdot (du, d\lambda, dy), \\ u_x \cdot du + \alpha dy. \end{aligned}$$

For the first term, we can compute $(D_u\phi_\tau du + D_\lambda\phi_\tau d\lambda)$ by evolution of equation (4.4.5). Specifically if we initialize the time-stepper for equation (4.4.5) with

$$(\hat{r}^+, \hat{c}^+, \hat{r}^-, \hat{c}^-, q_\ell, \widehat{r_w}^+, \widehat{c_w}^+, \widehat{r_w}^-, \widehat{c_w}^-, dq_\ell)$$

and evolve it to time τ. Let w^* be the solution after this evolution then we know

$$(D_u\phi_\tau du + D_\lambda\phi_\tau d\lambda) = (\widehat{r_{w^*}}^+, \widehat{c_{w^*}}^+, \widehat{r_{w^*}}^-, \widehat{c_{w^*}}^-)$$

and then this gives us a way to compute this term [13].

For the second term, we can compute $\Theta_y du$ very easily in Fourier space. Indeed for the more general $\Theta_y f$ we can state the explicit algorithm for the translational operator applied to f. We have from the Fourier transform,

$$\begin{aligned} f(x_j - y, t) &= \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} \hat{f}_k \phi_k(x_j - y), \\ &= \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} \hat{f}_k \phi_k(x_j)\phi_k(-y), \end{aligned}$$

and then we can see

$$\begin{aligned} \hat{f}_k\phi_k(-y) &= \hat{f}_k \exp(-iky), \\ &= \left(\widehat{r}_{f_k} + i\widehat{c}_{f_k}\right)(\cos(ky) - i\sin(ky)), \\ &= \left(\widehat{r}_{f_k}\cos(ky) + \widehat{c}_{f_k}\sin(ky)\right) + i\left(\widehat{c}_{f_k}\cos(ky) - \widehat{r}_{f_k}\sin(ky)\right), \end{aligned}$$

where $\hat{f}_k = \widehat{r_f}_k + i\widehat{c_f}_k$. So this gives us a way to explicitly compute the translation operator applied to a function easily in Fourier space. So $\Theta_y u$, Θu_x, and Θdu can all be computed with this algorithm as we know the Fourier space representations of u, u_x, and du. We mention that since dy is just a scalar, $\Theta_y u_x dy$ is no harder to compute. The next term, $T \cdot (du, d\lambda, dy)$ is already explicit as

$$T \cdot (du, d\lambda, dy) = T \cdot (\widehat{r_w}^+, \widehat{c_w}^+, \widehat{r_w}^-, \widehat{c_w}^-, dq_\ell, dy).$$

The term we deal with next is $u_x \cdot du$ and is slightly more involved to compute in Fourier space. We have that

$$u_x \cdot du = \langle u_x, du \rangle = \int_0^{2\pi} (u_x du)\, dx$$

and we can apply the Fourier transform to simplify and come up with an explicit formula for this as,

$$\begin{aligned}
\int_0^{2\pi} (u_x du)\, dx &= \sum_{j=0}^{N-1} \left(\left[\frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} (ik\hat{u}_k)\phi_k(x_j) \right] \left[\frac{1}{\sqrt{N}} \sum_{q=-\frac{N}{2}}^{\frac{N}{2}} \widehat{du}_q \phi_q(x_j) \right] \left[\frac{2\pi}{N} \right] \right), \\
&= \frac{2\pi}{N^2} \left(\sum_{k=-\frac{N}{2}}^{\frac{N}{2}} \sum_{q=-\frac{N}{2}}^{\frac{N}{2}} (ik\hat{u}_k \widehat{du}_q) \right) \left(\sum_{j=0}^{N-1} \phi_{k+q}(x_j) \right), \\
&= \frac{2\pi}{N^2} \left(\sum_{k=-\frac{N}{2}}^{\frac{N}{2}} \sum_{q=-\frac{N}{2}}^{\frac{N}{2}} (ik\hat{u}_k \widehat{du}_q) \right) (N\delta_{k,-q}), \\
&= \frac{2\pi}{N} \left(\sum_{k=-\frac{N}{2}}^{\frac{N}{2}} ik\hat{u}_k \widehat{du}_{-k} \right), \\
&= \frac{2\pi}{N} \left(\sum_{k=-\frac{N}{2}}^{\frac{N}{2}} ik\hat{u}_k \overline{\widehat{du}_k} \right).
\end{aligned}$$

Then let

$$\begin{aligned} S_k = ik\hat{u}_k\widehat{du_k} &= ik(\hat{r}_k + i\hat{c}_k)(\widehat{dr}_k - i\widehat{dc}_k), \\ &= ik\left[(\hat{r}_k\widehat{dr}_k + \hat{c}_k\widehat{dc}_k) + i(-\hat{r}_k\widehat{dc}_k + \hat{c}_k\widehat{dr}_k)\right], \\ &= k\left[(\hat{r}_k\widehat{dc}_k - \hat{c}_k\widehat{dr}_k) + i(\hat{r}_k\widehat{dr}_k + \hat{c}_k\widehat{dc}_k)\right], \end{aligned}$$

$S_k = R_k + iC_k$, and

$$\begin{aligned} u_x \cdot du &= \frac{2\pi}{N}\sum_{k=1}^{\frac{N}{2}}(S_k + S_{-k}), \\ &= \frac{2\pi}{N}\sum_{k=1}^{\frac{N}{2}}\left((R_k + iC_k) + (R_k - iC_k)\right), \\ &= \frac{4\pi}{N}\sum_{k=1}^{\frac{N}{2}} R_k. \end{aligned}$$

Therefore we have an explicit formula as

$$u_x \cdot du = \frac{4\pi}{N}\sum_{k=1}^{\frac{N}{2}-1} k(\hat{r}_k\widehat{dc}_k - \hat{c}_k\widehat{dr}_k)$$

where we have neglected $k = \frac{N}{2}$ because $R_{\frac{N}{2}} = 0$ since $\hat{c}_{\frac{N}{2}} = \widehat{dc}_{\frac{N}{2}} = 0$. The last term we have not dealt with is αdy but this is just scalar-scalar multiplication and is already explicit. With a way to compute each term in the matrix-vector product we now have the complete continuation method defined.

5.4 Results and more degeneracy

First we wish to test the corrector software by itself to ensure it is working correctly before doing any continuation. Figure 5.5 shows the approximate three bump equilibrium state we wish to correct. Figure 5.6 shows the result after the corrector has converged. One may notice the power spectrum of the corrected three bump equilibrium is only just beginning to decay exponentially and therefore is suspicious. There is a reason this may be happening. Figure 5.7 shows us the Newton residual and the GMRES residuals for every Newton iteration. The Newton residuals should decrease quadratically and certainly should be monotonic. Our Newton residuals however, do not decay

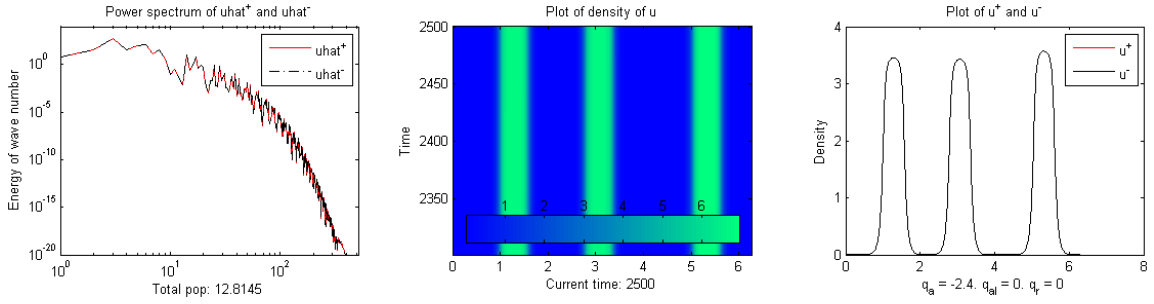

Figure 5.5: Power spectrum (left), total density plot through time (middle), and final time plot of density distributions (right) of the three bump equilibrium.

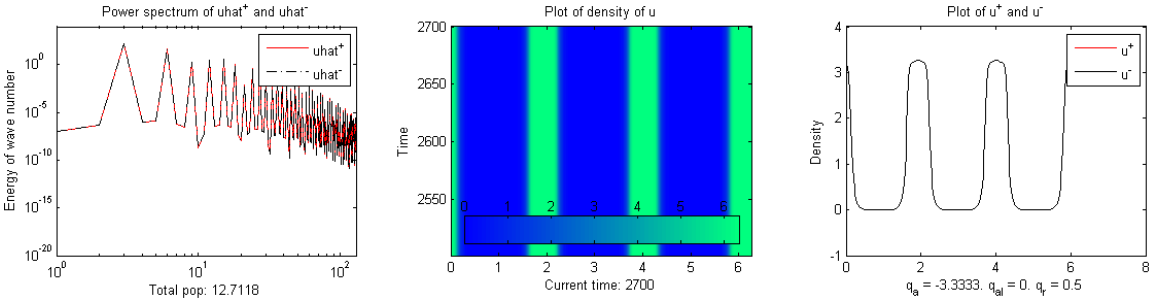

Figure 5.6: Power spectrum (left), total density plot through time (middle), and final time plot of density distributions (right) of the corrected three bump equilibrium.

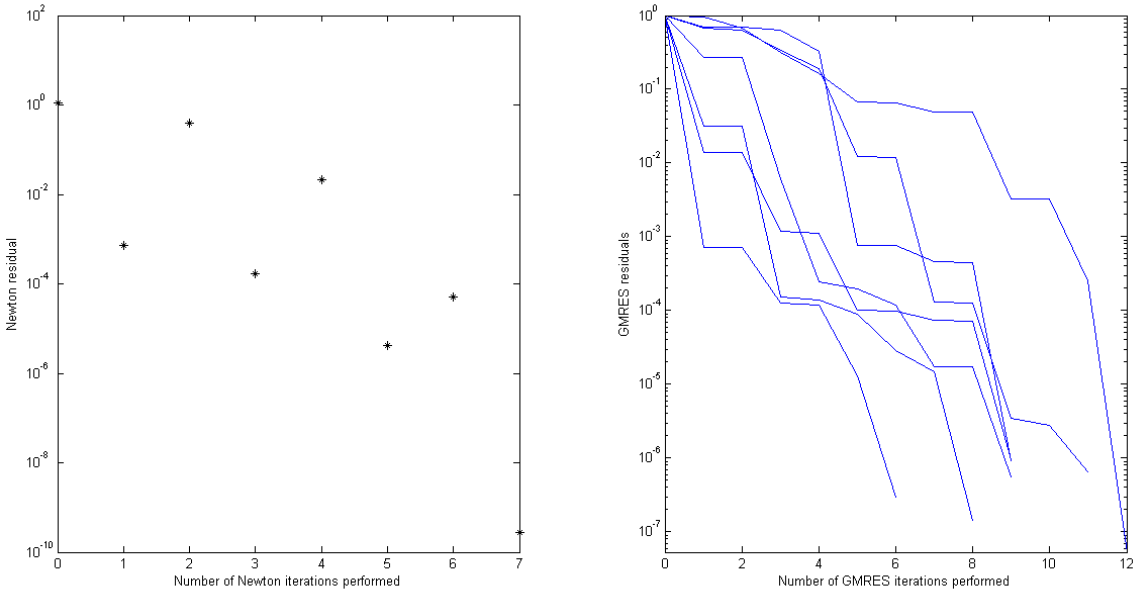

Figure 5.7: Newton residuals of the corrector algorithm applied to the three bump equilibrium (left) along with GMRES residuals for the solution of the Newton system on each update iteration (right).

quadratically and are not monotonic. While the GMRES residuals are reasonable, the Newton residuals are suspicious. Even worse, when the three bump equilibrium was corrected with tighter tolerances on the GMRES and Newton iterations the result was not a three bump equilibrium but was the homogeneous steady state. The corrector is clearly not working well.

Recent work has been done investigating the eigenvalues of the Newton system and more degeneracies were found. Figure 5.8 shows us seven eigenvalues and their associated eigenfunctions of $D\phi_\tau$. One of these should be the translational degeneracy we went over earlier and another seems to be coming from highest wave number, apparent from the high frequency eigenfunction. This degeneracy coming from the highest wave number should be able to be fixed with de-aliasing to remove at least the highest wave number. This still leaves five unaccounted for degeneracies. These have to be better understood before we begin to use the continuation algorithm we have.

In fact, we can remove one of the degeneracies. Figure 5.9 shows degenerate eigen-

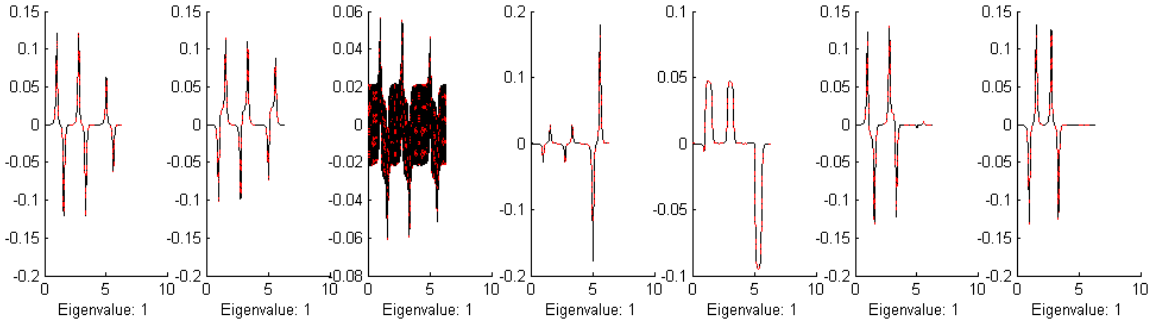

Figure 5.8: Seven degenerate eigenvalues of the Jacobian of the flow operator with their associated eigenfunctions.

values and eigenfunctions of a different three-bump equilibrium, different such that $N = 2^{11}$ for the solution and was computed independently of the other, with only six degenerate eigenvalues. This was achieved by de-aliasing half the spectrum, though only the highest wave number needs to be de-aliased in practice. This is because of the special treatment of the highest wave number in that the complex coefficient of the highest wave number must be zero. Any form of de-aliasing will remove this degeneracy as long as one is careful not to implicitly zero the "new" highest wave number. Thus we need not worry ourselves about this degeneracy.

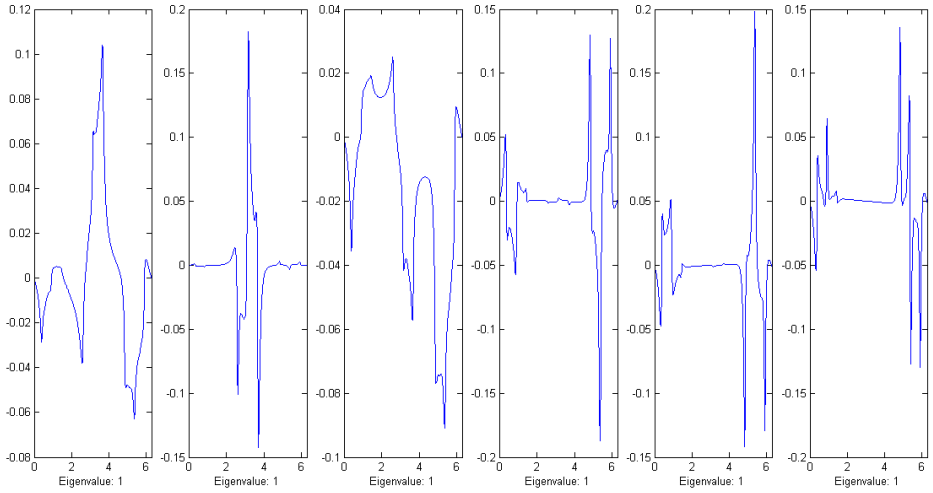

Figure 5.9: Six degenerate eigenvalues of the Jacobian of the flow operator with their associated eigenfunctions after performing de-aliasing to remove the degeneracy of the highest wave number. Quality of these eigenfunctions may be distorted because of the large de-aliasing applied in the test

Chapter 6
Conclusions and future work

At this point we have a time-stepper for equation (2.3.5) and equation (4.4.5). The simulations from our time-steppers match dynamics observed by Eftimie *et al.* [5], and all validation tests show that it is working as expected. However, for some parameter values the solutions show signs of a Gibb's phenomenon type error that might be arising. Furthermore the power spectrum of some of the solutions are not decaying exponentially, showing signs of aliasing errors. More research needs to be done into these errors.

We also have the theoretical setup as well as an implemented version of the continuation method. We thought we dealt with all the degeneracies in the Newton system and the corrector appeared to work for a three bump equilibrium but as was shown by the Newton residuals, something was going wrong. Indeed we discovered that there were other degeneracies in the Jacobian of the flow operator. Further work needs to be done in understanding these degeneracies and then in adding conditions to the Newton system or performing other fixes, such as de-aliasing at least the highest wave number, so that the updates to the solution avoid these degenerate directions.

Work is being done by examining the degenerate eigenvectors of some different states with fundamentally different structures as depicted in Figure 6.1. The number of degenerate eigenvalues varies depending on which solution we are considering and these degeneracies seem to have some relation to symmetries of the model which are also shared by the solution itself. Furthermore there appears to be a degeneracy caused by the highest wave number but we can fix this actually.

As for the next steps, after the continuation software is functioning correctly we can go about drawing out curves of states in parameter space. We will first start

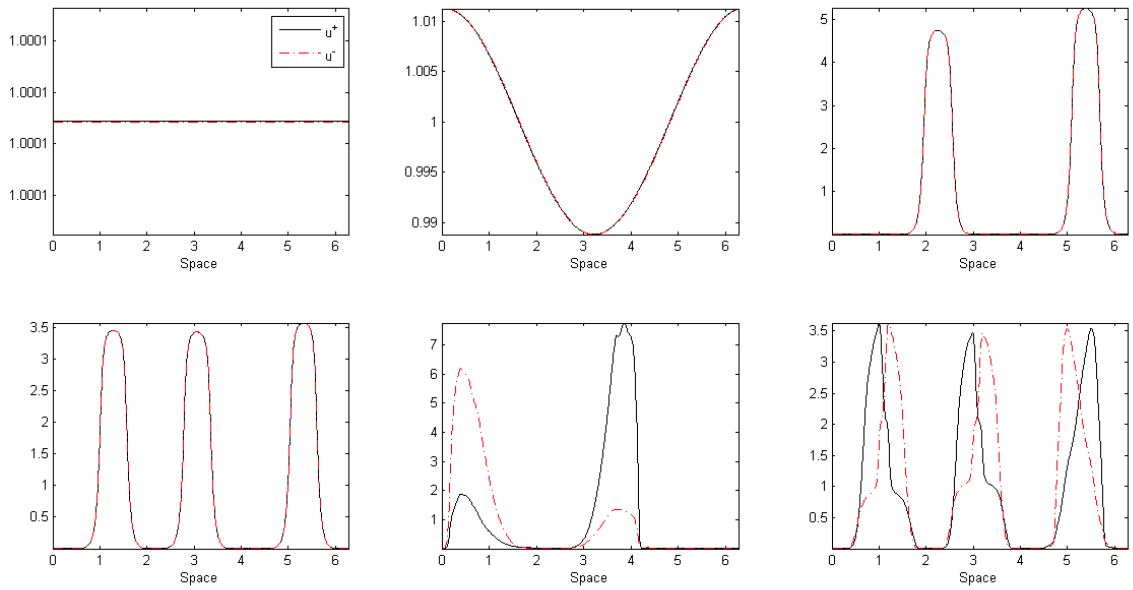

Figure 6.1: Homogeneous steady state (top left), one bump (top middle), two bump (top right), three bump (bottom left), double zigzag (bottom middle), and triple feather (bottom right).

drawing out curves from the homogeneous steady state point and compare results to linear analysis about the homogeneous steady state done by Eftimie *et al.* [11, 10], to further ensure things are working well. With that assurance along with other validation tests, such as looking at the Newton residuals, we can go about drawing out more curves of states and begin to get new results of how equation (2.3.5) depends on the parameters. We will then develop software to detect bifurcations as we draw out the curves so we can begin to draw boundaries in parameter space between regions of different observable dynamics by adding more conditions to the Newton system that will ensure our states points are also bifurcation points in parameter space.

Bibliography

[1] U. Ascher and L. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society of Applied and Industrial Mathematics, 1998.

[2] A. Bernoff and C. Topaz. A primer of swarm equilibria. *SIAM Journal on Applied Dynamical Systems*, 10:212–250, 2011.

[3] J. Brecht and D. Uminsky. Predicting pattern formation in particle interactions. *Mathematical Models and Methods in Applied Sciences*, 22, 2012.

[4] P-L. Buono and R. Eftimie. Codimension-2 bifurcations in animal aggregation models with symmetry. In preparation.

[5] R. Eftimie, G. de Vries, and M. Lewis. Complex spatial group patterns result from different animal communication mechanisms. *Proceedings of the National Academy of Sciences of the United States of America*, 104:6974–6979, 2007.

[6] R. Fetecau and A. Guo. A mathematical model for flight guidance in honeybee swarms. *Bulletin of Mathematical Biology*, 74:2600–2621, 2012.

[7] Y. Katz, K. Tunstrm, C. Ioannou, C. Huepe, and I. Couzin. Inferring the structure and dynamics of interactions in schooling fish. *Proceedings of the National Academy of Sciences of the United States of America*, 108:18720–18725, 2011.

[8] H. Keller. Numerical solution of bifurcation and nonlinear eigenvalue problems. *Applications of Bifurcation Theory*, pages 359–384, 1977.

[9] Y. Kuznetsov. *Elements of Applied Bifurcation Theory*, volume 112. Springer-Verlag New York, Inc., 2nd edition, 1998.

[10] M. Lewis, R. Eftimie, and G. de Vries. Weakly nonlinear analysis of a hyperbolic model for animal group formation. *SIAM Journal on Mathematical Biology*, 59:37–74, 2009.

[11] F. Lutscher, G. de Vries, M. Lewis, and R. Eftimie. Modeling group formation and activity patterns in self- organizing collectives of individuals. *Bulletin of Mathematical Biology*, 1537-1565, 2007.

[12] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.

[13] J. Sánchez, M. Net, B. García-Archilla, and C. Simó. Newton-Krylov continuation of periodic orbits for navier-stokes flows. *Journal of Computational Physics*, 11-33, 2004.

[14] C. Topaz, A. Bernoff, and A. Leverentz. Asymptotic dynamics of attractive-repulsive swarms. *Journal on Applied Dynamical Systems*, 8:880–908, 2009.

[15] L. Trefethen. *Spectral methods in MATLAB.* Society for Industrial and Applied Mathematics, 2000.